Umělé neuronové sítě v řízení systémů

Neural Networks in System Control

Bc. Ondřej Špaček

Diplomová práce
2017

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2016/2017

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ondřej Špaček**
Osobní číslo:	**A15174**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Automatické řízení a informatika**
Forma studia:	**prezenční**
Téma práce:	**Umělé neuronové sítě v řízení systémů**
Téma anglicky:	**Artificial Neural Networks in Systems Control**

Zásady pro vypracování:

1. **Vypracute literární rešerši z oblasti umělých neuronových sítí.**
2. **Zpracujte především využití umělých neuronových sítí v oblasti řízení systémů.**
3. **Implementujte vybranou umělou neuronovou síť.**
4. **Ověřte funkčnost regulace pomocí zvolené umělé neuronové sítě na vhodné soustavě.**
5. **Porovnejte regulaci neuronové sítě s jiným typem regulátoru.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **HAGAN, Martin T., Howard B. DEMUTH, Mark H. BEALE a Orlando DE JESÚS. Neural network design. 2st ed. Boston: PWS Pub., 2014. ISBN 9780971732117.**
2. **NORGAARD, Magnus, Ole RAVN, Niels K. POULSEN a L. K. HANSEN. Neural networks for modelling and control of dynamic systems: a practitioner's handbook. New York: Springer, 2000. ISBN 1852332271.**
3. **RASCHKA, Sebastian. Python Machine Learning. Birmingham: Packt Publishing, 2015. ISBN 9781783555130.**
4. **BOBÁL, Vladimír. Adaptivní a prediktivní řízení. Zlín: Univerzita Tomáše Bati, 2007. ISBN 978-80-7318-662-2.**
5. **ŠÍMA J., NERUDA R.: Teoretické otázky neuronových sítí. Vyd. 1. Praha, 1996: Matfyzpress, 390 s., ISBN 80-85863-18-9**
6. **KŘIVAN, Miloš. Úvod do umělých neuronových sítí. Vyd. 3., přeprac. Praha: Oeconomica, 2014, 44 s. ISBN 978-80-245-2024-7.**

Vedoucí diplomové práce: **doc. Ing. Zuzana Komínková Oplatková, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **3. března 2017**

Termín odevzdání diplomové práce: **22. května 2017**

Ve Zlíně dne 3. března 2017

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomové práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně 7. 5. 2017

.....

podpis autora

ABSTRAKT

Předmětem této práce je implementace vlastního regulátoru založeném na neuronové síti a následné porovnání oproti jiný běžným regulátorům. Implementovaný algoritmus řízení umělou neuronovou sítí je otestován jak v simulaci v programu Simulik, tak na reálném modelu v laboratoři. Teoretická část se zabývá umělými neuronovými sítěmi, jejich použitím v identifikací a v regulaci. Úvod praktické části je věnován implementaci neuronové sítě v MATLAB. Další část se zabývá porovnáním regulátorů s regulátory tvořenými umělými neuronovými sítěmi. V závěrečné kapitole je ukázka aplikace na reálném modelu.Výsledkem této práce je implementovaný regulátor využívající neuronové sítě a porovnání regulátorů tvořených umělými neuronovými sítěmi s běžnými regulátory. Porovnání v této práci jsou provedeny z hledisek počtu matematických operací, velikosti potřebné paměti pro výpočet, integrálních regulačních kritérií, relativních překmitů, minim a maxim zásadních veličin regulačního pochodu.

Klíčová slova: Umělé neuronové sítě, Regulátor tvořený umělou neuronovou sítí, Regulace, Simulace, Identifikace, MATLAB, Simulink

ABSTRACT

The main subject of this master thesis is an implementation of artificial neural network controller and subsequently comparison of other typical controllers, with implemented artificial neural network controller. Algorithm of artificial neural network is tested in simulation and on a real model in the laboratory. Theoretical part of this thesis is about artificial neural networks and their uses in system identification a control. In the practical part of this thesis is started with implementation of neural network in MATLAB. Follows comparison of different controllers with artificial neural network. In the last chapter is an example implementation of artificial neural network controller on real system. The result of this thesis is an implemented artificial neural network controller and comparatively with different common controllers. Comparisons are based on the number of operations, size of memory, integral control criteria, overshoot minima and maxima of control response.

Keywords: Artificial neural networks, Artificial neural network controller, Regulation, Simulation, Identification, MATLAB, Simulink

Rád bych poděkoval vedoucí diplomové práce paní doc. Ing. Zuzaně Komínkové Oplatkové, Ph.D. za odborné vedení a připomínky při zpracování této práce.

Děkuji také mé rodině za podporu během studia.

OBSAH

ÚVOD

Mnoho schopností, které v dnešní době lidé ovládají se naučili na příkladech. Naučili jsme se létat jako ptáci, potápět se hluboko do moří jako ryby, naučili jsme se jak vyrábět stroje podobné naší konstrukci, které nás nahradí při monotónní a těžké práci. Je tedy logické, že se lidé pokouší napodobit funkci mozku algoritmy neuronových sítí, které jsou vhodné pro řešení mnohých úloh, kde standardní metody selžou.

Jedním z prvních použití neuronových sítí bylo rozeznávání vzorů v obrazu. Od té doby jsou neuronové sítě silně spjaty s počítačovým viděním a pronikly do mnoha jiných disciplín jako je předvídání počasí, automatické řízení vozidel, do lékařství nebo právě řízení systémů a mnoho dalších.

Typickým úkolem v řízení systémů je návrh regulátoru a identifikace. Tento postup nelze realizovat bez specifikace popisu chování řízeného systému, to ovšem neplatí u neuronových sítí. Metody s využitím umělých neuronových sítí jsou výhodné hlavně z důvodu, že vyžadují jen minimum apriorních informací a o řízeném systému uvažují jako o „*černé krabici*“. Jsou proto velmi často využívané pro řízení složitých nelineárních systémů s více vstupy a výstupy nebo s jedním vstupem a výstupem, jako jsou chemické reaktory, roboty a stíhací letadla.

Umělé neuronové sítě se silně promítly do teorie řízení nejsilněji do oblasti prediktivního řízení. Důvodem jsou umělé neuronové rekurentní sítě, které se pomocí zpětné vazby dokážou chovat dynamicky a velmi dobře predikovat budoucí hodnoty. Další oblast využití jsou běžné regulátory, kde se využívá přímo vazebních umělých neuronových sítí. Tyto regulátory našly uplatnění jako vylepšení nelineárních regulátorů nebo jejich náhrada. V neposlední řadě umožňují neuronové sítě použití časově variantního adaptivního řízení, kdy se využívá přímo vazební nebo rekurentní umělá neuronová síť, která je naučena online.

Hlavním cílem praktické části této práce je implementace vlastního regulátoru založeném na umělé neuronové síti. Ověření tohoto algoritmu na reálném systému a porovnání s jinými běžnými běžnými typy regulátorů.

Cílem teoretické části je seznámení čtenáře s problematikou umělých neuronových sítí, seznámení se způsoby návrhu regulátorů a identifikace umělou neuronovou sítí.

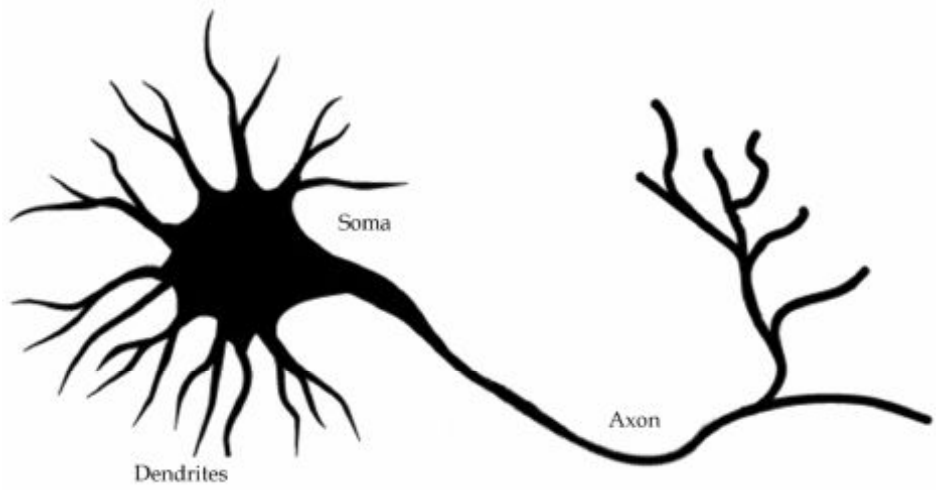
I. TEORETICKÁ ČÁST

1 Umělé neuronové sítě

Jedná se o typ matematického modelu, často nelineárního, který je tvořený několika umělými neurony. Tento model využívá učení s učitelem, tzn. jsou známy vstupní vzorky a jim odpovídající cíle učení. Tyto modely vznikly jako snaha napodobení biologických neuronových sítí.

1.1 Umělý neuron

Biologické neurony jsou buňky propojené nervy. V mozku jsou součástí přenosu a zpracování informací s využitím chemických a elektrických signálů. Prvním zjednodušeným modelem biologického neuronu byl McCulloc-Pittův neuron (dále jen MCP).

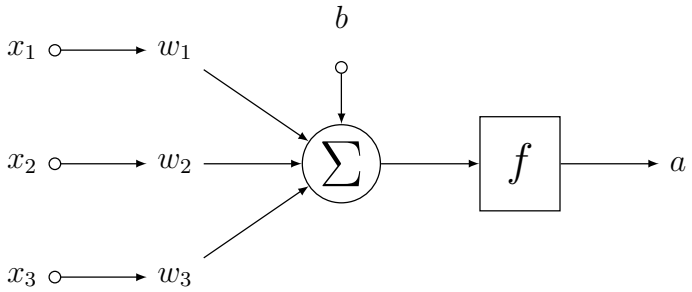


Obr. 1.1. Biologický neuron

Pánové McCulloc a Pitt popsali biologický neuron jako logické hradlo s binárním výstupem. Několik signálů vstupuje do umělého neuronu a pokud akumulovaný signál překročí určitý práh, bude výstupní signál 1 v opačném případě 0. Zásadním problémem bylo, že neexistovalo pravidlo pro nalezení vah, které by umožnily vstupní informace převést na požadovaný výstupní stav. To se ovšem změnilo v roce 1957 kdy pan Rosenbatt navrhl učící pravidlo pro neuron vycházející z MCP neuronu. Tento neuron nazval Perceptron a jeho učící pravidlo umožňovalo za určitých podmínek naleznout optimální velikost vah $w_1, w_2, ...w_n$, které násobí vstupy $x_1, x_2, ...x_n$ na základě součtu těchto součinů a prahu b je rozhodnuto o výstupu a pomocí aktivační funkce $f(n)$, která v případě perceptronu byla lineární $f(n) = n$. Od té doby došlo k velkému pokroku neuronových sítí, ale tento pohled na neuron zůstal prakticky stejný [3].

1.1.1 Matematický model neuronu

Jak již bylo řečeno obecný neuron je tvořen vahami $w_1, w_2, ...w_n$, prahem b a aktivační funkcí $f(n)$ [5]. Schéma neuronu, které ukazuje jak mezi sebou tyto parametry interagují je na následujícím obrázku



Obr. 1.2. Schéma neuronu

Platí tedy (1.1).

$$n = x_1 \cdot w_1 + x_2 \cdot w_2 + ... + x_n \cdot w_n + b \tag{1.1}$$

Pro výstup z neuronu platí (1.2).

$$a = f(n) \tag{1.2}$$

Kde x_r je vstup do r-té synapse, w_r je váha pro r-tou synapsi, b je práh neuronu, n je výstup sumace neuronu, $f(n)$ je aktivační funkce a a – výstup neuronu.

Prostudováním těchto rovnic lze vytvořit vektorový zápis ve tvaru (1.3), pro sumaci součinu vah a vstupů.

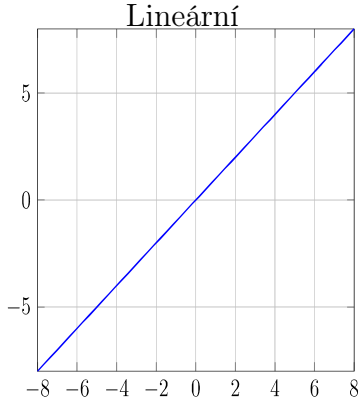
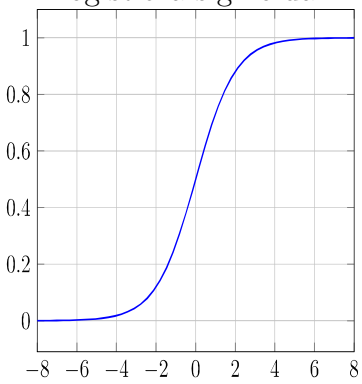
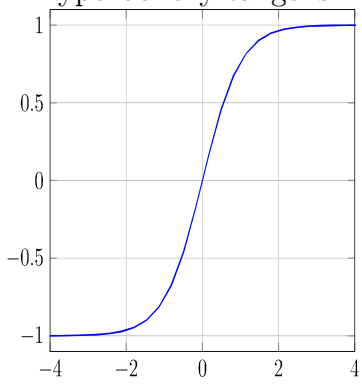
$$n = \begin{bmatrix} w_1 & w_2 & ... & w_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + b = \boldsymbol{w}^T \boldsymbol{x} + b \tag{1.3}$$

Kde potom $\boldsymbol{w}$ označuje vektor vah a $\boldsymbol{x}$ vektor vstupů

1.1.2 Aktivační funkce

Se znalostí matematického modelu neuronu lze předpokládat, že aktivační funkce bude mít ústřední funkci v neuronu. Její volbou lze ovlivnit jak obor hodnot výstupu neuronu, tak současně ovlivníme to, na jaké skupiny problémů lze neuron použít. V tabulce (Tab. 1.1) jsou uvedeny pouze vybrané aktivační funkce, které jsou vhodné pro aplikace v oblasti řízení. Je potřeba poznamenat, že aktivační funkce umělé neuronové sítě musí být diferencovatelná [6].

Tab. 1.1. Aktivační funkce

Graf aktivační funkce	Aktivační funkce	Použití
Lineární 	$f(x) = x$ $\frac{df(x)}{dx} = 1$	Tato funkce se označuje jako **identity**. Výstup neuronových sítí, kde se kontinuálně mění výstupní hodnota např. predikce. $f(x) \in \langle -\infty nebo \infty \rangle$
Logistická sigmoida 	$f(x) = \frac{1}{1+e^{-x}}$ $\frac{df(x)}{dx} = f(x)(1-f(x))$	Tato funkce se označuje jako **logsig**, nebo **logistic**. Výstup neuronových sítí, kde je výstupem pravděpodobnost, nebo nelineární funkce skryté vrstvy neuronové sítě. $f(x) \in \langle 0, 1 \rangle$
Hyperbolický tangens 	$f(x) = \frac{2}{1+e^{-2x}} - 1$ $\frac{df(x)}{dx} = 1 - f(x)^2$	Tato funkce se označuje jako **tansig**, nebo **tanH**. Nelineární skrytá vrstva, neuronové sítě. $f(x) \in \langle -1, 1 \rangle$

1.1.3 Kritérium

Kritérium určuje jak se bude vyhodnocovat cíl učení, typicky kvadrát rozdílu výstupu neuronové sítě a cíle. Existuje velké množství kritérií vhodných pro učení neuronových sítí, ale v řízení se nejčastěji používá střední kvadratické kritérium. Chybu e lze určit pomocí rovnice (1.4), jako rozdíl vektoru cílů $\boldsymbol{t}$ a vektoru výstupu neuronu pro všechny vzorky $\boldsymbol{a}$ (rozměr obou vektorů je $N \times 1$).

$$\boldsymbol{e} = \boldsymbol{t} - \boldsymbol{a} \tag{1.4}$$

Střední kvadratické kritérium je $F(x) = E\left[\boldsymbol{e}^T\boldsymbol{e}\right]$ a jeho odhad lze zapsat pomocí rovnice (1.5).

$$\hat{F}(x) = \frac{1}{N}\boldsymbol{e}^T\boldsymbol{e} = \frac{1}{N}\sum_{i=1}^{N}[t(i) - a(i)] \tag{1.5}$$

Největší výhodou tohoto kritéria je, že je velmi jednoduché [1].

1.1.4 Učení neuronu

Nejdůležitějším problémem je zřejmě to, jak naučit neuron aby podle vstupních vzorků sledoval své cíle. Již bylo řečeno, že zásadní vliv na učení má hlavně aktivační funkce a vyhodnocovací kritérium.

Jak tedy postupovat při určení optimálních vah neuronu? Je zřejmé, že jediné parametry, které máme lze modifikovat pro zlepšení hodnoty kritéria, jsou váhy $\boldsymbol{w}$ a práh b. Pokud je známá funkce od vstupu do neuronu až k výstupu z kritéria, lze ji zderivovat podle $\boldsymbol{w}$, b. Výsledkem bude gradient funkce, který můžeme použít k optimalizaci vah a prahu libovolnou metodou. Při práci s neuronovými sítěmi se často používá řetízkového pravidla, které problém rozdělí na subproblémy, což umožňuje jednoduché změny aktivačních funkcí a kritérií. Pokud je známa funkce $F(x) = f(g(x))$ potom lze její derivaci nalézt pomocí rovnice (1.6).

$$\frac{\partial F(x)}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x} \tag{1.6}$$

Při úvaze umělé neuronové sítě s aktivační funkcí $f(\boldsymbol{w}, b, \boldsymbol{X})$,kritériem $F(\boldsymbol{w}, b, \boldsymbol{X}, a)$, matici všech vstupních vzorků sítě $\boldsymbol{X}$ a vektor cílů $\boldsymbol{t}$. Derivace lze učit podle (1.7) a (1.8).

$$\frac{\partial F(\boldsymbol{w}, b, \boldsymbol{X}, a)}{\partial \boldsymbol{w}} = \frac{\partial F(\boldsymbol{w}, b, \boldsymbol{X}, a)}{\partial f(\boldsymbol{w}, b, \boldsymbol{X})} \cdot \frac{\partial f(\boldsymbol{w}, b, \boldsymbol{X})}{\partial n(\boldsymbol{w}, b)} \cdot \frac{\partial n(\boldsymbol{w}, b)}{\partial \boldsymbol{w}} \tag{1.7}$$

$$\frac{\partial F(\boldsymbol{w}, b, \boldsymbol{X}, a)}{\partial b} = \frac{\partial F(\boldsymbol{w}, b, \boldsymbol{X}, a)}{\partial f(\boldsymbol{w}, b, \boldsymbol{X})} \cdot \frac{\partial f(\boldsymbol{w}, b, \boldsymbol{X})}{\partial n(\boldsymbol{w}, b)} \cdot \frac{\partial n(\boldsymbol{w}, b)}{\partial b} \tag{1.8}$$

Příkladem odvození pravidla pro trénování neuronu s lineární aktivační funkcí obecně označovaný jako ADELINE. Při úvaze středního kvadratického kritéria $F(\boldsymbol{w}, b, x, a) = \frac{1}{N}\boldsymbol{e}^T\boldsymbol{e}$, aktivační funkci $\boldsymbol{a} = f(\boldsymbol{w}, b, \boldsymbol{X})$. Derivace kritéria podle aktivační funkce, je určena v rovnici (1.9).

$$\frac{\partial F(\boldsymbol{w}, b, \boldsymbol{X}, a)}{\partial f(\boldsymbol{w}, b, \boldsymbol{X})} = \frac{\partial}{\partial f(\boldsymbol{w}, b, \boldsymbol{X})} \frac{1}{N} (\boldsymbol{t} - \boldsymbol{a})^T (\boldsymbol{t} - \boldsymbol{a}) = \frac{2}{N} (\boldsymbol{a} - \boldsymbol{t})^T \tag{1.9}$$

Derivace aktivační funkce podle sumy součinu vstupů s váhami a prahu je určena v rovnici (1.10).

$$\frac{\partial f(\boldsymbol{w}, b, \boldsymbol{X})}{\partial n(\boldsymbol{w}, b)} = \frac{\partial n(\boldsymbol{w}, b)}{\partial n(\boldsymbol{w}, b)} = 1 \tag{1.10}$$

Derivace sumy součinu vstupů s váhami a prahu podle vah (1.11) a prahu (1.12) ($\vec{\mathbf{1}}$ symbolizuje jednotkový vektor který umožní ke každému vzorku přičíst práh)

$$\frac{\partial n(\boldsymbol{w}, b)}{\partial b} = \frac{\partial (\boldsymbol{w}^T \boldsymbol{X} + \vec{\mathbf{1}} \cdot b)}{\partial b} = \vec{\mathbf{1}} \tag{1.11}$$

$$\frac{\partial n(\boldsymbol{w}, b)}{\partial \boldsymbol{w}} = \frac{\partial (\boldsymbol{w}^T \boldsymbol{X} + \vec{\mathbf{1}} \cdot b)}{\partial \boldsymbol{w}} = \boldsymbol{X}^T \tag{1.12}$$

Při derivacích je nutné myslet na rozměr výsledku (derivace matic mají dvě řešení původní tvar a transpozice). Výsledný gradient může tedy být (1.13), nebo **??**.

$$\begin{aligned} \frac{\partial F^T(\boldsymbol{w}, b, \boldsymbol{x}, a)}{\partial \boldsymbol{w}} &= \frac{2}{N} (\boldsymbol{a} - \boldsymbol{t}) \boldsymbol{X}^T \\ \frac{\partial F^T(\boldsymbol{w}, b, \boldsymbol{x}, a)}{\partial b} &= \frac{2}{N} (\boldsymbol{a} - \boldsymbol{t}) \vec{\mathbf{1}}^T \end{aligned} \tag{1.13}$$

nebo

$$\begin{aligned} \frac{\partial F(\boldsymbol{w}, b, \boldsymbol{x}, a)}{\partial \boldsymbol{w}} &= \frac{2}{N} \boldsymbol{X} (\boldsymbol{a} - \boldsymbol{t})^T \\ \frac{\partial F(\boldsymbol{w}, b, \boldsymbol{x}, a)}{\partial b} &= \frac{2}{N} \vec{\mathbf{1}} (\boldsymbol{a} - \boldsymbol{t})^T \end{aligned} \tag{1.14}$$

Výsledný gradient lze použít k optimalizaci libovolnou optimalizační metodou.

1.2 Matematický model vrstvy umělé neuronové sítě

Vrstva neuronové sítě je tvořena několika umělými neurony, které mohou ale nemusí být kompletně propojeny, tzn. umělá neuronová vrstva s dvěma vstupy a dvěma neurony muže mít propojení oba vstupy na oba neurony, každý vstup na vlastní neuron nebo jeden vstup na oba neurony a druhý na jeden. Propojení neuronů se stejně jako v biologii nazývá synapse. Synapse mezi neurony mohou být různé, ale minimálně při prvních učení sítě se používá plného propojení všech vstupů na všechny neurony dané vrstvy. Následně je pomocí různých metod možné eliminovat nějaké synapse, tím zrychlit přímý průchod a přetrénování neuronové sítě. Každý neuron může mít svou aktivační funkci, ale typicky má každá vrstva stejnou aktivační funkci pro zjednodušení učení umělých neuronových sítí. Matematicky lze problém vyjádřit pomocí rovnice (1.15).

$$\boldsymbol{n} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b} \tag{1.15}$$

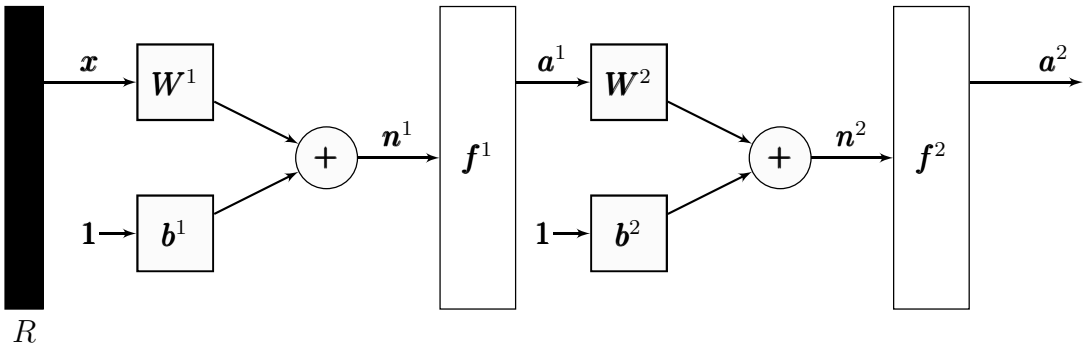
Učení jedené vrstvy je prakticky velmi podobné učení jednoho neuronu.

1.3 Vícevrstvý perceptron

Vícevrstvý perceptron je umělá neuronová síť tvořená několika vrstvami, která byla navržena pro nalezení řešení nelineárně separabilních problémů [6]. První vrstva neuronové sítě se nazývá vstupní vrstva, jejím úkolem je propojit vstupy a neurony další vrstvy, která se nazývá skrytá. Účelem skryté vrstvy je uvést silnou nelinearitu v podobě aktivační funkce. Skrytých vrstev může být neomezeně mnoho, ale nejčastěji se u vícevrstvého perceptronu používá jedna. Výstupní vrstva se skládá z požadovaného počtu výstupů neuronové sítě a aktivační funkce vhodné pro daný problém. Matematicky se jedná o určitý počet navzájem propojených vrstev, pro jednu skrytou vrstvu lze získat rovnici (1.16).

$$\boldsymbol{a^2} = \boldsymbol{f}^2(\boldsymbol{W}^2\boldsymbol{f}^1(\boldsymbol{W}^1\boldsymbol{x} + \boldsymbol{b}^1) + \boldsymbol{b}^2) \tag{1.16}$$

Typicky se umělá neuronová síť zobrazuje ve formě různých grafů například jako na obrázku (Obr. 1.3).



Obr. 1.3. Schéma neuronové sítě

1.3.1 Backpropagation

Hledaní gradientu vzhledem k vahám vrstev vzdálených od výstupní vrstvy je poměrně složité. Z toho důvodu se využívá obecný přístup, který se nazývá backpropagation. S algoritmem backpropagation je možné nalézt gradienty jakékoliv přímo vazební neuronové sítě. Principem tohoto algoritmu je počítaní gradientů od výstupní vrstvy a postupná propagace chyby výstupu směrem ke vstupní vrstvě [1].

Pro nalezení optimálních vah je nutné znát vstupní vzorky $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ a požadované cíle učení $\{\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_n\}$. Prvním krokem je aplikace vstupních vzorků na vstup umělé neuronové sítě a vypočtení chyby na výstupu pomocí zvoleného kritéria $\boldsymbol{f}(\boldsymbol{a}^n)$. Z trénování jednoho neuronu jsou známy rovnice (1.8), (1.7) pokud jsou upraveny do

obecného tvaru, kde m symbolizuje vrstvu sítě jsou získány rovnice (1.17).

$$\begin{aligned}\frac{\partial \boldsymbol{F}(\boldsymbol{a}^m)}{\partial \boldsymbol{W}^m} &= \frac{\partial \boldsymbol{F}(\boldsymbol{a}^m)}{\partial \boldsymbol{f}^m(\boldsymbol{n}^m)} \cdot \frac{\partial \boldsymbol{f}^m(\boldsymbol{n}^m)}{\partial \boldsymbol{n}^m(\boldsymbol{W}^m, \boldsymbol{b}^m)} \cdot \frac{\partial \boldsymbol{n}^m(\boldsymbol{W}^m, \boldsymbol{b}^m)}{\partial \boldsymbol{W}^m} \\ \frac{\partial \boldsymbol{F}(\boldsymbol{a}^m)}{\partial \boldsymbol{b}^m} &= \frac{\partial \boldsymbol{F}(\boldsymbol{a}^m)}{\partial \boldsymbol{f}^m(\boldsymbol{n}^m)} \cdot \frac{\partial \boldsymbol{f}^m(\boldsymbol{n}^m)}{\partial \boldsymbol{n}^m(\boldsymbol{W}^m, \boldsymbol{b}^m)} \cdot \frac{\partial \boldsymbol{n}^m(\boldsymbol{W}^m, \boldsymbol{b}^m)}{\partial \boldsymbol{b}^m}\end{aligned} \tag{1.17}$$

Z rovnic je patrné, že ve výstupní vrstvě bude výpočet jednoduchý ovšem pro vrstvy od $m-1$ nebude známý člen $\frac{\partial \boldsymbol{F}(\boldsymbol{a}^{m-1})}{\partial \boldsymbol{f}^{m-1}(\boldsymbol{n}^{m-1})}$ a zde nastoupí algoritmus backpropagation. Pro nalezení algoritmu backpropagation je třeba zavést rekurentní vztah mezi vrstvami pomocí (1.18) Jacobiho matice.

$$\frac{\partial \boldsymbol{n}^{m+1}}{\partial \boldsymbol{n}^m} = \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \frac{\partial n_1^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_1^{m+1}}{\partial n_N^m} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} & \frac{\partial n_2^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_2^{m+1}}{\partial n_N^m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial n_{N+1}^{m+1}}{\partial n_1^m} & \frac{\partial n_{N+1}^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_{N+1}^{m+1}}{\partial n_N^m} \end{bmatrix} \tag{1.18}$$

Protože je derivace lineární operace, tak opět pomocí řetízkového pravidla lze nalézt člen $\frac{\partial \boldsymbol{F}(\boldsymbol{a}^m)}{\partial \boldsymbol{f}^m(\boldsymbol{n}^m)}$ v jiné než výstupní vrstvě platí rovnice (1.19).

$$\frac{\partial \boldsymbol{F}(\boldsymbol{a}^m)}{\partial \boldsymbol{f}^m(\boldsymbol{n}^m)} \cdot \frac{\partial \boldsymbol{f}^m(\boldsymbol{n}^m)}{\partial \boldsymbol{n}^m} = \left(\frac{\partial \boldsymbol{n}^{m+1}}{\partial \boldsymbol{n}^m}\right)^T \frac{\partial \boldsymbol{F}(\boldsymbol{a}^m)}{\partial \boldsymbol{n}^{m+1}} \tag{1.19}$$

Součin členů $\frac{\partial \boldsymbol{F}(\boldsymbol{a}^m)}{\partial \boldsymbol{f}^m(\boldsymbol{n}^m)} \cdot \frac{\partial \boldsymbol{f}^m(\boldsymbol{n}^m)}{\partial \boldsymbol{n}^m}$ se nazývá citlivost a bude značen $\boldsymbol{s}^m$. Dalším krokem je zobecnění výpočtu Jacobiánu (1.18), kde studiem jednoho prvku této matici lze určit maticový výpočet (1.20) (i značí řádek a j sloupec matice)[1].

$$\frac{\partial n_j^{m+1}}{\partial n_i^m} = \frac{\partial \left(\sum_{t=1}^{L^N} w_{j,t}^{m+1} a_t^m + b_j^{m+1}\right)}{\partial n_i^m} = w_{j,i}^{m+1} \frac{\partial a_i^m}{\partial n_i^m} = w_{j,i}^{m+1} \frac{\partial f_i^m(n_i^m)}{\partial n_i^m} \tag{1.20}$$

V maticovém tvaru lze zapsat rovnici do podoby (1.21).

$$\frac{\partial \boldsymbol{n}^{m+1}}{\partial \boldsymbol{n}^m} = \boldsymbol{W}^{m+1} \frac{\partial \boldsymbol{f}^m(\boldsymbol{n}^m)}{\partial \boldsymbol{n}^m} \tag{1.21}$$

Kde matice derivací funkcí je uvedena v rovnici (1.22).

$$\frac{\partial \boldsymbol{f}^m(\boldsymbol{n}^m)}{\partial \boldsymbol{n}^m} = \begin{bmatrix} \frac{\partial f_1^m(n_1^m)}{\partial n_1^m} & 0 & \cdots & 0 \\ 0 & \frac{\partial f_2^m(n_2^m)}{\partial n_2^m} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \frac{\partial f_3^{L^N}(n_{L^N}^m)}{\partial n_{L^N}^m} \end{bmatrix} \tag{1.22}$$

Pro doplnění takto vypadá matice derivací funkcí pro všechny běžné aktivační funkce, kromě normalizovaných výstupních funkcí jako je SoftMax, kde místo nul budou další derivace. Tato matice také často bývá redukována na vektor a maticové násobení nahrazeno násobením po prvcích (bývá značeno $\odot$). Výsledná citlivost pro výstupní vrstvu je určena rovnicí (1.23).

$$\boldsymbol{s}^m = \frac{\partial \boldsymbol{f}^m(\boldsymbol{n}^m)}{\partial \boldsymbol{n}^m} \left(\boldsymbol{W}^{m+1}\right)^T \frac{\partial \boldsymbol{F}(\boldsymbol{a}^m)}{\partial \boldsymbol{n}^{m+1}} \tag{1.23}$$

Pro vnitřní vrstvy potom platí rovnice (1.24).

$$\boldsymbol{s}^m = \frac{\partial \boldsymbol{f}^m(\boldsymbol{n}^m)}{\partial \boldsymbol{n}^m} \left(\boldsymbol{W}^{m+1}\right)^T \boldsymbol{s}^{m+1} \tag{1.24}$$

Gradient pro jakoukoliv vrstvu lze určit pomocí rovnic (1.25) a (1.26).

$$\frac{\partial \boldsymbol{F}(\boldsymbol{a}^m)}{\partial \boldsymbol{W}^m} = \boldsymbol{s}^m \frac{\partial \boldsymbol{n}^m(\boldsymbol{W}^m, \boldsymbol{b}^m)}{\partial \boldsymbol{W}^m} \tag{1.25}$$

$$\frac{\partial \boldsymbol{F}(\boldsymbol{a}^m)}{\partial \boldsymbol{b}^m} = \boldsymbol{s}^m \frac{\partial \boldsymbol{n}^m(\boldsymbol{W}^m, \boldsymbol{b}^m)}{\partial \boldsymbol{b}^m} \tag{1.26}$$

1.4 Rekurentní neuronové sítě

Rekurentní neuronové sítě jsou typem umělých neuronových sítí. Tyto sítě obsahují alespoň jednu zpětnou vazbu. Často budou obsahovat také zpoždění pro signály, potom se označují jako dynamické neuronové sítě. Protože výstupy jsou závislé na vstupech současných a také minulých, lze tento typ umělé neuronové sítě použít k aproximaci dynamických jevů. Ve výsledku takové neuronové sítě působí jako typ prediktoru několik kroků do budoucna. Tyto neuronové sítě mohou být trénovány standardními algoritmy, ale gradienty jsou závislé na čase, proto je třeba ve výpočetním algoritmu s tímto počítat.

V dnešní době jsou používany dvě metody pro určení rekurentních neuronových sítí. První metoda je obdoba backpropagace a nazývá se backpropagace skrz čas. Tato metoda je výpočetně méně náročná než následující, ale nelze ji implementovat online. Důvodem je, že tento algoritmus funguje zpětně v čase tzn. musíme začít z poslední hodnotou po ukončení měření. Druhou metodou je real-time rekurentní učení tato metoda je výpočetně značně složitější ale umožňuje implementaci online. V případě kdy je třeba počítat jakobián tak tato metoda je výhodnější než backpropagace skrz čas [1].

Principem obou těchto metod je rozvinutí rekurentní neuronové sítě do podoby kdy jde o čistě dopřednou umělou neuronovou síť. V případě, že jsou známy 3 vzorky dat a neuronové síť o jedné vrstvě má jednu zpětnou vazbu bude neuronová síť při učení odpovídat tří vrstvé dopředné umělé neuronové síti.

Kromě výpočetní složitosti mají obě metody další problémy jako zmenšování velikosti gradientu. Dalším zásadním problémem je, že povrch na kterém je prováděna optimalizace bude obsahovat oblasti které nejsou spojeny se skutečnou dynamikou aproximovaného systému, ale jedná se čistě o oblasti kde je rekurentní neuronová síť nestabilní. Toto může způsobit problém použitým optimalizačním metodám. Dalším probléme je, že vstupy musí být použity v sekvenci v jaké mají být rekurentní neuronovou sítí naučeny. Je tedy mnohem důležitější, než u statických sítí, dosáhnout veškerých možností v trénovací množině [1].

2 Učení neuronových sítí neuronových sítí

K učení neuronových sítí lze využít prakticky jakoukoliv optimalizační metodu. Používané metody lze rozdělit na numerické a numerické, které mají nějaký heuristický faktor. Velmi častým optimalizačním algoritmem umělých neuronových sítí je metoda největšího spádu a její modifikace.

2.1 Metoda největšího spádu (gradient descent)

Tato metoda patří k jedené z nejjednodušších a nejznámějších v numerické optimalizaci. Vychází ze standardního pravidla pro numerickou optimalizaci uvedeného v rovnici (2.1).

$$x_{k+1} = x_k + \alpha d \tag{2.1}$$

Kde α je učící konstanta a d je směr, pro hledání minima. Principem je požadavek na zmenšení funkce v dalším kroku, tedy $F(x_{k+1}) < F(x_k)$. Za předpokladu, že je uvažován rozvoj funkce okolo staré hodnoty x_k v Taylorovu řadu jako v rovnici (2.2).

$$F(x_{k+1}) = F(x_k + \Delta x) \approx F(x_k) + \nabla F(x)\Big|_{x=x_k} \Delta x \tag{2.2}$$

Protože je požadavek k minimalizaci tak $\nabla F(x_k)\Delta x < 0$. Je tedy zřejmé, že α bude malé kladné číslo a směr největšího spádu se získá změnou znaménka gradientu. Výsledné pravidlo pro minimalizaci touto metodou je rovnice (2.3).

$$x_{k+1} = x_k - \alpha \nabla F(x_k) \tag{2.3}$$

Kde $\nabla F(x_k)$ je gradient funkce.

Důvodem velké oblíbenosti této metody v oblasti umělých neuronových sítí je, že není potřeba určovat Hessovu matici, úlohu lze vektorizovat, nebo paralelizovat pro výpočet na GPU. Nevýhodou této metody je její nestabilita při velké učící konstantě, nebo velmi pomalá konvergence při konstantě malé. Další nevýhodou je, že při backpropagation se gradient zmenšuje, proto bez modifikace tuto metodu neleze použít na hluboké neuronové sítě.

2.1.1 Adaptivní učící se koeficient

Jedna z nejjednodušších metod upravující délku gradientu pracuje na jednoduchém principu principu, když se zvětší kritérium oproti minulému kroku o určitou hranici, což by naznačovalo oscilace okolo minima, zmenší se učící koeficient. Častá volba hranice kdy se zmenšuje gradient je 5%. Naopak pokud se sníží kritérium potom dojde ke zvětšení koeficientu. V případě pokud je použit moment a kritérium je menší než v minulém kroku, tak se nastaví na nulu do doby dokud nedojde v dalším kroku ke snížení kritéria.

2.1.2 Moment

Moment je jednoduchá heuristická modifikace heuristických metod založených na největším spádu. Existují v zásadě dvě metody první označovaná běžně jako moment (2.5) a druhá jako Nestorův urychlený gradient (2.6), nebo Nestorův moment. Cíl těchto metod je určitým způsobem stabilizovat směr gradientu [1]. Tato metoda působí jako filtr směru gradientu a ten nemůže rychle měnit svůj směr. K filtraci gradientu se používá předchozí přírůstek vah Δw a koeficient označující moment η.

$$\Delta x_k = -\alpha \nabla F(x_{k-1}) + \eta \Delta x_{k-1} \tag{2.4}$$

Zapsané do normálního tvaru, bude proměnná změna jako

$$x_{k+1} = x_k - \alpha \nabla F(x_k) + \eta \Delta x_k \tag{2.5}$$

Nebo Nestorův urychlený gradient, který navíc využívá diference gradientů a muže se použít i další koeficient momentu λ, nebo $\lambda = \eta$.

$$x_{k+1} = x_k - \alpha \nabla F(x_k) + \alpha\lambda(\nabla F(x_k) - \nabla F(x_{k-1})) + \eta \Delta x_k \tag{2.6}$$

Metody založené na momentu značně urychlují učení heuristickými metodami. Mají ovšem i nevýhodu oproti neheuristickým metodám a to, že se zvyšuje počet parametrů, které je třeba určit metodou pokus omyl.

3 Inteligentní regulace

Inteligentní řízení je odvětví regulace, které se zabývá využitím matematických aparátů, které ukazují někdy lepší přístup než běžné metody. Mezi teorie inteligentního řízení patří umělé neuronové sítí, Bayesovská statistiky, Fuzzy teorie a Neuro-Fuzzy teorie [11]. Následující kapitoly jsou věnovány využití neuronových sítí v oblasti regulace.

3.1 Experimentální identifikace

Při návrhu regulátoru je často nutné znát model systému a v případě, kdy je získání modelu fyzikálními a matematickými možnostmi velmi složité, je vhodné použít identifikaci. Velkou nevýhodou je že většina teorie věnující se identifikaci je pouze lineární. Mezi hlavní důvody využití lineární teorie je jednoduchost výpočtů, jednoduší návrh regulátoru a některé nelineární systémy jde vhodně aproximovat lineárním modelem [13].

V praxi se ale často vyskytují velmi silné nelinearity a proto zavedením nelineárního modelu lze typicky získat velké zlepšení, zde se tedy objevuje možnost umělých neuronových sítí v identifikaci.

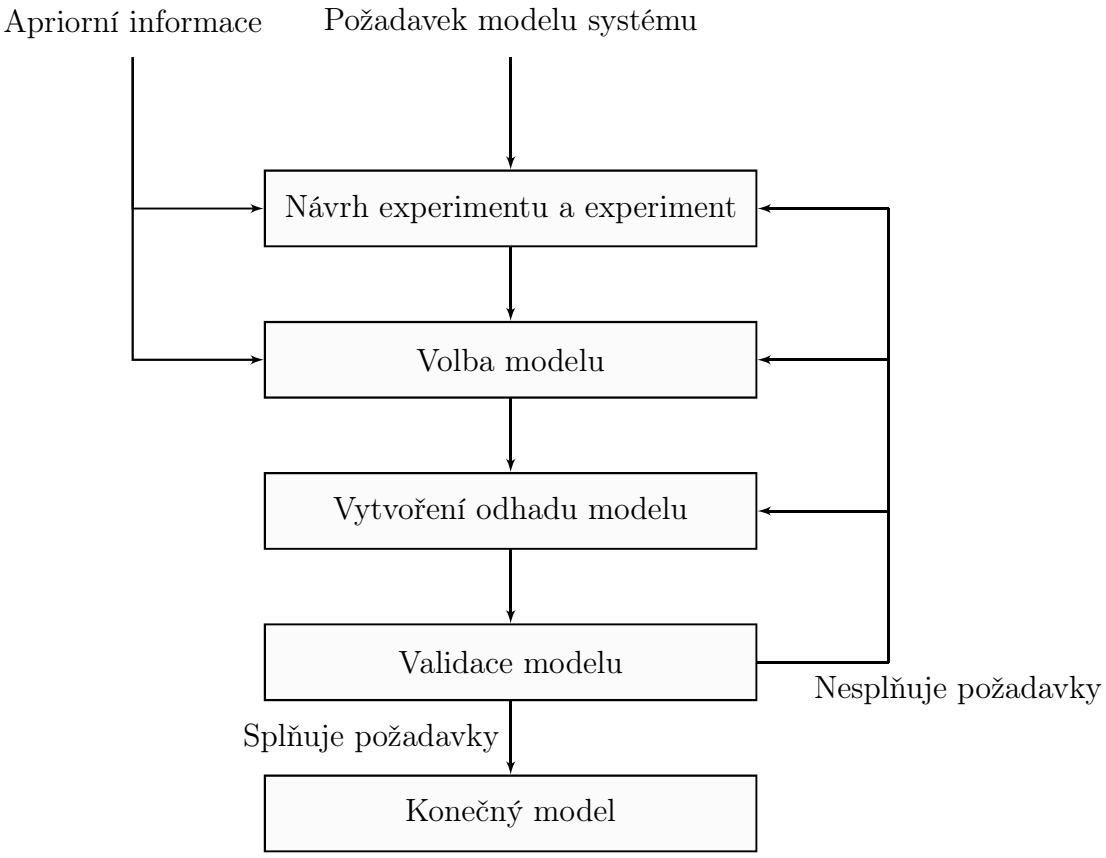
3.1.1 Postup při experimentální identifikaci

Postup při experimentální identifikaci se skládá z několika kroků, při kterých je možné použít předem známé informace o systému (tzv. apriorní informace).

Prvním krokem bývá **návrh experimentu a experiment**. Prvotní experiment by nám měl odhalit základní chování systému, pracovní oblast a jeho nelinearity, tyto může být apriorní informace. Dále by se mělo pokračovat návrhem finální podoby experimentu, která bude sloužit ke sběru dat, která budou použita k identifikaci. Při návrhu experimentu je třeba zvolit vhodný identifikační signál, vzorkovací periodu a pracovní oblast [13]. Naměřená data jsou dále zpracovávána metodami, které umožní odstranit poruchy ze signálu a šum. Následně mohou být data použita v další fázi identifikace. V případě identifikace nestabilních systémů je nutné nejdříve stabilizovat zpětnou vazbu regulátorem.

Dalším krokem v proceduře experimentální identifikace je **volba modelu**. Často se jedná o více stupňovou volbu, kdy můžeme zvolit lineární přístup a následně model ARX, nebo umělou neuronovou síť a model NARX. Tato volba může být ovlivněna apriorními informacemi o systému.

Následně získáme **odhad skutečného modelu**. V této fázi podle zvoleného kritéria vybíráme ze skupiny modelů, které byly zvoleny v předchozím kroku. Nejtypičtějším kritériem je chyba výstupu, kdy chybu lze určit jako rozdíl skutečného výstupu a od-



Obr. 3.1. Běžný postup identifikace

hadu výstupu. Mezi další používané modely patří chyba vstupu, kdy je model inverzí identifikovaného modelu a kritérium je minimální chyba mezi vstupem modelu a odhadem vstupu. V neposlední řadě se používá chyba rovnice, kde se využije jak přímého modelu tak inverzního modelu k systému. **Validace modelu** je poslední fází a jedná se o zjištění zda se podařilo dosáhnout předem daných požadavků na identifikaci. Pokud se to nepodařilo je třeba se vrátit k předchozím krokům experimentální identifikace.

3.1.2 Identifikační modely využívající neuronové sítě

Nelineární modely využívající umělé neuronové sítě jsou silně inspirovány modely lineárními vycházejícími z obecné rovnice stochastického modelu (3.1).

$$y(t) = G(z^{-1})u(t) + H(z^{-1})e_s(t) \tag{3.1}$$

V lineárním stochastickém modelu vystupuje přenos systému $G(z^{-1})$ a přenos působení stochastické poruchy $H(z^{-1})$. Tento model lze v lineárním případě popsat v rovnici (3.2) pomocí vektoru parametrů Φ a regresního vektoru $\Theta(k)$.

$$\hat{y}(k) = \Theta^T(k)\,\Phi + e(k) \tag{3.2}$$

V případě kdy $G(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})}z^{-d}$ a $H(z^{-1}) = \frac{1}{A(z^{-1})}z^{-d}$ je získán jeden z nejznámějších modelů používaným pro identifikaci a to sice model ARX (AutoRegressive,

eXenogenous input). Tento model využívá regresního vektoru (3.3)

$$\Theta(k) = [y(k-1), \ldots, y(k-n), u(k-d) \ldots, u(k-d-m)]^T \tag{3.3}$$

Vektoru parametrů se potom dá vyjádřit rovnicí (3.5).

$$\Phi = [a_1, \ldots, a_n, b_0 \ldots, b_m]^T \tag{3.4}$$

V případě použití nelineární neuronové sítě, jako je vícevrstvý perceptron, která bude sloužit pro učení se nelineární závislosti mezi vstupy a výstupy sítě, se volby modelu redukují pouze na volbu počtu vstupů do sítě a volba vnitřní struktury sítě. Vstupy se velmi často volí právě jako ve formě lineárních modelů. Tento přístup má několik výhod, jako je např. jednoduchá úprava vnitřní struktury sítě, logické návaznost na znalosti lineárních systémů a vhodnost pro návrh regulátoru. Nelineární rovnice modelu je vyjádřena ve tvaru (3.5).

$$\hat{y}(k) = g[\Theta(k)] + e(k) \tag{3.5}$$

Funkce g v tomto modelu reprezentuje nelineární funkci přímého průchodu neuronovou sítí. Typické označení nelineárních modelů je pomocí počátečního písmena N. Tedy pokud použijeme stejný regresní vektor jako u lineárního modelu ARX potom nelineární model bude NARX [2].

Identifikační modely využívající neuronové sítě lze dále rozdělit na ty co budou vždy stabilní a na ty co mohou být nestabilní. Toto je velmi důležité hlavně u nelineárních systémů kde se stabilita určuje těžko při přístupu "black box".

Nejtypičtěji používaným modelem neuronové sítě, který je vždy stabilní z důvodu, že nemá žádnou zpětnou vazbu je NARX. Popis modelu NARX využívá stejný regresní vektor jako ARX. Dalším stabilním identifikačním modelem je NFIR (Nonlinear Finite Impulse Response), který lze v určitých případech využít k dobré aproximaci systému i když neuvažuje minulé hodnoty výstupu. Jedná se tedy o nelineární FIR filtr. Regresní vektor NFIR je uveden v rovnici (3.6).

$$\hat{y}(k) = g[u(k-d) \ldots, u(k-d-m)] \tag{3.6}$$

Někoho by mohlo napadnout, proč používat modely, které mohou být nestabilní, důvodem je, že mají zpětnou vazbu (implementace tedy vyžaduje umělé rekurentní neuronové sítě). Tato zpětná vazba umožňuje provádět predikci mnoho kroků dopředu. Typickým příkladem je model NARX, uveden v rovnici (3.7), kdy se skutečné hodnoty nahradí predikovanými hodnotami, jako v (3.7). V případě modelu NARX se zpětnou vazbou se někdy používá název NNOE (Neural Network Output Error) [2].

$$\hat{y(k)} = g[\hat{y}(k-1), \ldots, \hat{y}(k-n), u(k-d) \ldots, u(k-d-m)] \tag{3.7}$$

3.1.3 Vhodné signály pro identifikaci

Jedním z nejdůležitějších faktorů při správné identifikaci umělou neuronovou sítí je volba identifikačního signálu. Zatím co v lineární teorii jsou oblíbené PRBS (pseudo

random binary sequence) a použití několika sinusoid s různou frekvencí [13]. Tak v nelineárních případech je to složitější. Ideálně by se měly použít všechny frekvence se všemi možnými amplitudami a stejnosměrnými složkami signálu. Takovéto řešení je ale často velmi nereálné a zdlouhavé. Možnou náhradou v tomto ohledu je použití frekvenčního rozmítání o několika amplitudách a stejnosměrných složek signálu. V zásadě se jedná o sinus se zvětšující se frekvencí (nebo zmenšující se frekvencí). Tento signál lze popsat rovnicí 3.8.

$$x(k) = x_0 + A\sin\left(\omega_{k_0} kT + \frac{k^2}{N}[\omega_{k_1} - \omega_{k_0}]T\right) \tag{3.8}$$

kde ω_{k_0} značí počáteční frekvenci, ω_{k_1} konečnou frekvenci a x_0 stejnosměrnou složku signálu.

Ovšem jednoznačně nejpoužívanější funkcí pro identifikaci s umělými neuronovými sítěmi je funkce která se skládá ze skoků o náhodném počtu opakovaní, matematicky lze vyjádřit rovnicí 3.9. Tato funkce bývá označována jako „skyline“ (podobá se mrakodrapům) nebo „random walk“. Důvodem popularity této funkce je že zjednodušuje volbu vzorkovací frekvence, pro dobrý popis systému jsou k dispozici jak dlouhé úseky, kdy se signál může ustálit, ale i rychlé změny, pro dobrou identifikaci dynamických stavů [2].

$$x(k) = x(k-1) + e(t|\mu, \sigma^2) \tag{3.9}$$

Kde e symbolizuje filtrovaný bílý šum, σ^2 je rozptyl a μ je průměrná hodnota. Při výpočtu je rozhodnuto který z členů bude zvolen podle pravděpodobnosti, uvedené v podmínkách 3.10.

$$x(k) = \begin{cases} x(k-1) & P(x(k) = x(k-1)) = \alpha \\ e(t|\mu, \sigma) & P(x(k) = e(t|\mu, \sigma^2)) = 1 - \alpha \end{cases} \tag{3.10}$$

3.1.4 Příprava dat

Příprava dat hraje v identifikaci hlavní roli a obzvláště u neuronových sítí, kde je třeba dbát na více věcí. První fází by měl být průzkum spektra dat a návrh filtru, který eliminuje šumy. Toto je velmi zásadní krok zvláště u neuronové sítě, která používá skutečné výstupy a skutečné vstupy. U takových to modelů, bez filtru skutečných dat, lze jen velmi špatně dosáhnout kvalitního modelu. Velmi jednoduchým a zároveň dobrým typem filtru je Butterworthův filtr, který má ve frekvenční odezvě rovnou část téměř až po lomovou frekvenci. Pro případ dolní propusti, což bude nejpoužívanější filtr při filtrování dat, je vyjádřen rovnicí 3.11.

$$H(s) = \frac{k_0}{\prod_{k=1}^{n} \frac{(s-s_k)}{\omega_c}} \tag{3.11}$$

Kde k_0 je zesílení při $\omega = 0$, ω_c je zlomová frekvence a s_k je pól, pro který platí $s_k = \omega_c e^{\frac{j(2k+n-1)}{2n}}$. Jmenovatele pro několik prvních řádů jsou dostupné v různých

tabulkách v normalizované formě, stačí tedy dosadit do normalizovaného polynomu $s_n = \frac{s}{\omega_c}$, výpočet je tedy velmi jednoduchý [12].

Dalším krokem je eliminace redundantních dat. Při trénování neuronových sítí a využití odhadu kvadratického kritéria musí být uvažováno, že všechny možné vstupy neuronové sítě jsou rovnoměrně zastoupeny. Pokud tomu tak nebude, tak chyba v oblasti kde bylo více hodnot bude minimalizována více než v oblasti, kde bylo hodnot méně. V některých případech tento výsledek ale může být přínosný, například když řízení bude prováděno hlavně v jedné části pracovní oblasti, je zde možné použít více dat.

Posledním krokem typickým pro neuronovou síť je normalizace dat. Při pohledu na různé aktivační funkce, je patrné, že jsou omezeny (kromě lineární). Je tedy třeba tomu přizpůsobit data, při úvaze že $logsig(5) = 0,9933$ a $logsig(6) = 0,9975$, kde relativní rozdíl je jen $0,42\%$. Naopak ve velmi nelineární oblasti $logsig(0,1) = 0,525$ a $logsig(0,2) = 0,5498$ což je relativní rozdíl $4,72\%$. Je tedy zřejmé, že u této funkce je výhodnější normalizovat data do oblasti -1 až 1, protože to umožní síti lépe využít nelinearitu aktivační funkce. Důvodem normalizace výstupu je to, že když je požadavek rozsahu větší než -1 až 1 , musí být tento rozsah získán na úkor dalších neuronů jejichž přítomnost by pouze zpomalovala učení (např. s jedním výstupním umělým neuronem s logsig nelze dosáhnou větší hodnoty než 1).

3.1.5 Délka vektoru zpoždění

Jedním z velkých problémů při identifikaci je vhodné určení délky vektoru zpoždění. Zatím co v lineární teorii je jednoduše známé, že pro systém druhého řádu je třeba znát poslední dva vstupy a poslední dva výstupy u nelineárních toto neplatí. V případě volby nevodně velkého vektoru se ztrácí výpočetní čas, dále je ztížen nebo úplně znemožněn návrh regulátoru s umělou neuronovou sítí s tímto modelem a také můžou nastat jiné matematické problémy. Pokud je naopak zvolen příliš krátký vektor potom dojde ke špatné identifikaci dynamických stavů. K nalezení optimální délky vektoru lze využít předpoklad Lipschitzovy kontinuity funkce [2]. Při předpokladu modelu, který je popsán $y(k) = g[\Theta(k)]$, kde $\Theta(k) = [y(k-1), \ldots, y(k-n), u(k-d) \ldots, u(k-d-m)]^T$. Při předpokladu že l^2 norma derivací systému podle regresních vektorů je omezená [14], potom platí rovnice 3.12.

$$\|f\| = \left\| \frac{\partial g}{\partial \Theta} \right\| < C \tag{3.12}$$

Potom Lipschitzův kvocient lze nalézt, pro $i \neq j$ pomocí rovnice 3.14.

$$q_{ij} = \left\| \frac{y(k_i) - y(k_j)}{\Theta(k_i) - \Theta(k_j)} \right\| < C \tag{3.13}$$

Tento výpočet je třeba provést pro každou volbu regresního vektoru se zpožděním n, z důvodu velké složitosti výpočtu se většinou uvažuje zpoždění vstupů a výstupů jako

stejné. Následně se vybere m největších kvocientů, což by mělo být okolo cca 1% z celého souboru vypočtených kvocientů. Následuje vyhodnocení kritéria podle rovnice 3.14.

$$Q_n = \left(\prod_{k=1}^{m} \sqrt{n} q_n(k) \right)^{\frac{1}{m}} \tag{3.14}$$

Po vynesení kritéria do grafu se najde bod, kde dochází ke největšímu zlomu a ten určuje optimální délku regresního vektoru.

3.1.6 Vyhodnocení kvality neuronové sítě v identifikaci a regulaci

K vyhodnocení kvality neuronové sítě je poměrně mnoho nástrojů, ale ani jeden jednoznačně neřekne, že je síť velmi dobrá, nebo špatná, až na základě několika ukazatelů lze o kvalitě rozhodnout.

Typickým a zároveň velmi jednoduchým ukazatelem pro neuronové sítě je pozorování vývoje kritéria v několika množinách. Prvním krokem je tedy vhodné rozdělení dat nejčastěji na tři skupiny. První skupina budou data na které, bude síť trénována tj. trénovací množina. Druhou skupinou bude testovací množina, jejíž účel je ověření kvality modelu a používá se pro nastavení volitelných parametrů sítě při testování. Poslední částí dat je validační množina, tato množina může sloužit několika úkolům jako je předčasné ukončení trénování, pokud kritérium validační množiny stoupá po několik kroků. Další možností využití je použití k úplně finální validaci sítě. Při použití validační množiny je důležité, aby nebyla používána jako testovací. Jinak to má za následek že je snaha dosáhnout co nejlepšího výsledku v testovací i validační množině a to prakticky vede k tomu že se spojí do jedné množiny. Výsledkem je potom špatná generalizace neznámých dat [1].

Mezi možnosti vyhodnocení výstupů patří lineární regrese mezi skutečnými výstupy a cíli učení. Tuto metodu lze použít pro případy, kdy jsou data prokládané funkcí umělé neuronové sítě. Regrese je vyjádřena rovnicí 3.15 a její koeficienty lze vypočítat pomocí 3.16.

$$\hat{y} = at + b + \epsilon \tag{3.15}$$

Kde $\hat{y}$ je skutečný výstup, t je cíl učení, ϵ je residuální chyba, koeficient a je směrnice a koeficient b je posunutí.

$$\hat{a} = \frac{\sum_{k=1}^{N}(t_k - \bar{t})(y_k - \bar{y})}{\sum_{k=1}^{N}(t_k - \bar{t})^2} \tag{3.16}$$

Kde $(t_k - \bar{t})$ je odhad směrnice,t_k je k-tý cíl a $\bar{t}$ je průměr cílů. Odhad parametru posunutí lze provést za předpokladu $\epsilon = 0$ v 3.15. Korelační koeficient R lze vypočítat pomocí rovnice 3.17.

$$R = \frac{\sum_{k=1}^{N}(t_k - \bar{t})(y_k - \bar{y})}{(N-1)\sigma_t \sigma_y} \tag{3.17}$$

Kde σ_t, $sigma_y$ jsou směrodatné odchylky cílů a výstupů sítě. V případě že R nebo R^2

se blíží jedničce jedná se o dobré proložení funkce neuronovou sítí. V případě nízkých hodnot R je možné často spatřit důvod, jako je velký počet vzdálených hodnot. Tato analýza by měla být prováděna na každé množině dat samostatně, tak i na všech množinách současně [1].

Dalším velmi jednoduchým ukazatelem je vynesení chyb $e = t - \hat{y}$ do histogramu. Touto metodou lze potvrdit, že existují vzdálené body při učení a možná je bude nutno odstranit. Jejich odstraněním a přetrénováním celé sítě lze zásadně snížit finální hodnotu trénovacího kritéria.

V případě, že výstup je umělé neuronové sítě je závislý na čase, je třeba tuto závislost zkontrolovat. A to v případech i s rekurentní sítí i bez rekurentní sítě. Ke kontrole v této oblasti lze využít korelační funkce. Běžně se zkoumá autokorelační funkce chyby 3.18.

$$\hat{R}_{ee} = \frac{1}{N-\tau} \sum_{t=1}^{N-\tau} e(t)e(t+\tau) \tag{3.18}$$

Pokud budou chyby nezávislé, tak by autokorelační funkce měla ideálně odpovídat autokorelační funkci bílého šumu. To je pulz v nule a jinde nula. Protože se pracuje pouze s odhadem tak se uvažuje místo nuly rozsah $\pm 5\%$. Pro $\tau \neq 0$ musí tedy platit rovnice 3.19.

$$-\frac{2\hat{R}_{ee}(0)}{\sqrt{N}} < \hat{R}_{ee} < \frac{2\hat{R}_{ee}(0)}{\sqrt{N}} \tag{3.19}$$

Pokud je tedy výsledná korelační funkce v tomto intervalu lze považovat chybu za nekorelovanou. Další zkoumanou funkcí je korelační funkce mezi sekvencí vstupů a chyb 3.20. V případě korelační funkce by neměla nastat žádná závislost mezi sekvencemi, tedy ani v čase $\tau = 0$.

$$\hat{R}_{xe} = \frac{1}{N-\tau} \sum_{t=1}^{N-\tau} x(t)e(t+\tau) \tag{3.20}$$

Kde x je posloupnost vstupů. Interval 3.21, kde je vstup a chyba považována za nekorelovanou je $\pm 5\%$ pro všechna τ.

$$-\frac{2\sqrt{\hat{R}_e(0)}\sqrt{\hat{R}_x(0)}}{\sqrt{N}} < \hat{R}_{xe} < \frac{2\sqrt{\hat{R}_e(0)}\sqrt{\hat{R}_x(0)}}{\sqrt{N}} \tag{3.21}$$

V zásadě lze využít jakoukoliv korelační funkci pro ověřování sítě, ale dvě zmíněné jsou hlavní. Důvodem je, že je nežádoucí aby chyba v budoucnosti byla korelována s chybou v minulosti. Stejně tak je špatné pokud je jakkoliv korelován vstup s chybou, to je ukazatel nedostatečné extrakce informací z dat. Zlepšení korelačních funkcí lze většinou dosáhnout zvětšením vektoru zpoždění vstupů.

3.2 Regulátory

Hlavní účel regulátorů s neuronovou sítí je v současné teorii řízení a regulace nalezení řídícího zákona pro neznámé/složité nelineární systémy. Metody návrhu regulátoru

využívajícího neuronových sítí se dělí do dvou kategorií na přímé a nepřímé. Přímá metoda slouží k přímému navržení regulátoru z naměřených dat. Implementace těchto metod bývá většinou jednoduchá, ale požadované chování regulátoru se dosahuje jen velmi těžko. Při každé změně parametrů je potřeba přetrénovat celou neuronovou síť. Nepřímá metoda využívá identifikovaného modelu pomocí neuronové sítě, který je pak využit k navržení regulátoru. Implementace nepřímých metod bývá poměrně složitá a trénování může být velmi dlouhé. Výhodou některých metod spadajících do této kategorie, je že není třeba přetrénovat neuronovou síť, při změně parametrů. Nepřímé metody patří k velmi často používaným v reálné praxi .

3.2.1 Přímé inverzní řízení

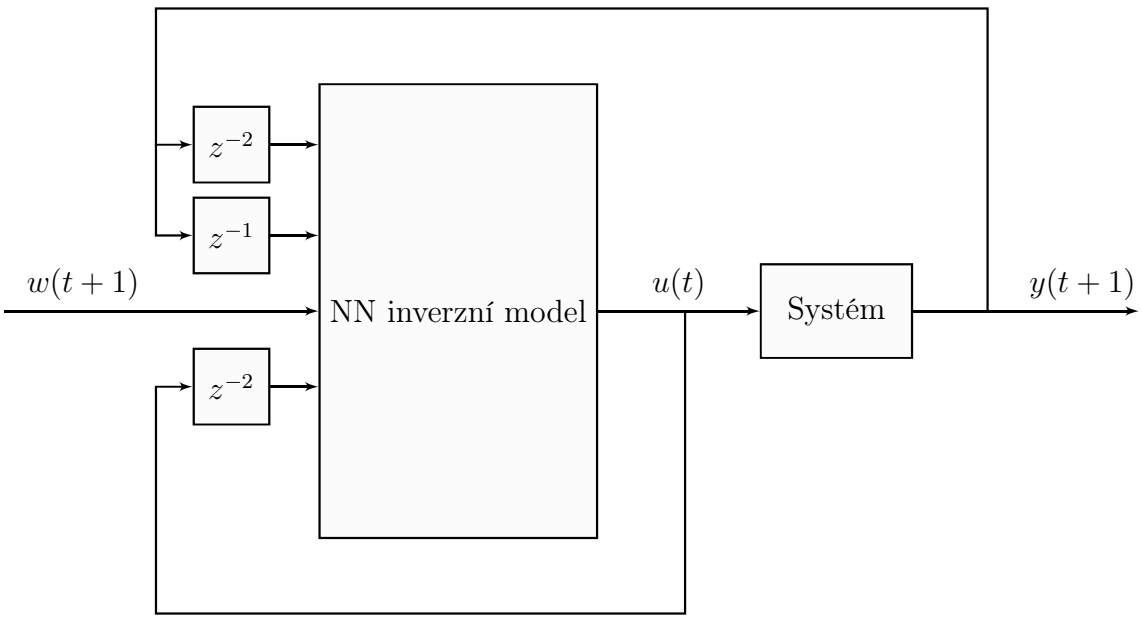
Jednou z prvních metod, které byly navrženy pro řízení nelineárních systémů je přímé inverzní řízení. Princip je velmi jednoduchý s cílem trénovat umělou neuronovou síť jako inverzi systému. Pokud tedy lze najít inverzi systému, tak ji lze použít jako regulátor [2]. Při předpokladu nelineárního diskrétního systému popsaného rovnicí (3.22).

$$y(t+1) = g[y(t), \ldots, y(t-n+1), u(t), \ldots, u(t-m)] \tag{3.22}$$

Potom odhad hledané inverze lze zapsat jako (3.23).

$$\hat{u}(t) = \hat{g}^{-1}[y(t+1), y(t), \ldots, y(t-n+1), u(t), \ldots, u(t-m)] \tag{3.23}$$

Za předpokladu existence inverze, může být tato umělá neuronová síť využita k řízení využitím substituce $w(t+1)$ za $y(t+1)$. Princip je schématicky naznačený na obrázku (Obr. 3.2).



Obr. 3.2. Regulátor prvního řádu

K nalezení inverze se používají dva typy trénování obecné a specializované. Obecné

trénování využívá jednoduchého principu, které je velmi podobné identifikaci ovšem využívající regresoru (3.23).

Obecné trénovaní využívá kritérium k trénovaní sítě v podobě (3.25).

$$J = \frac{1}{2N}\sum_{t=1}^{N}[u(t) - \hat{u}(t)]^2 \tag{3.24}$$

Velkou výhodou je, jak už bylo řečeno, že není třeba mít znalost matematického modelu. Další výhodou je, že tuto metodu lze použít offline. Z předchozích rovnic lze usoudit že přenos řízení bude ve tvaru (3.25)

$$G_{w/y}(z^{-1}) = z^{-1} \tag{3.25}$$

Je tedy zřejmé, že tato metoda generuje podobný regulátor jako je dead-beat a v principu se jedná o to, že regulátor linearizuje systém a žádaná hodnota je sledována se zpožděním jednoho, popřípadě více kroků.

Tyto regulátory mají velmi špatnou robustnost a velkou citlivost na poruchy. Pokud je inverze nestabilní, tak celá regulační smyčka bude nestabilní. Inverze může být nestabilní i v případě, že spojitý model je stabilní. Dalším problémem, je že pokud pro dva různé akční zásahy je dosaženo stejného stavu, potom inverzní model neexistuje. Důležitou věcí, je vhodný identifikační signál, který by v tomto případě měl reprezentovat signál, který bude použitý jako žádaný.

3.2.2 Nepřímé inverzní řízení

Dalším možným přístupem k trénování inverzního modelu je nepřímá metoda nazývající se specializované trénování. Při obecném trénování se často dosáhne velmi dobrého modelu, který ale nebude moc dobrý v generalizaci neznámých dat. Výsledný regulátor nebude vždy dokonalý a může mít stálou regulační odchylku, jak v ustáleném stavu tak i při dynamických jevech. Důvodem je to, že optimalizace není cílená, to znamená, že není optimalizovaný přímo výstup. Odpovědí na tyto problémy je specializované trénování které minimalizuje rozdíl mezi žádanou hodnotou a jedná se tedy o cílenou optimalizaci. Žádaná hodnota může také být upravena referenčním modelem (3.26), pro potřeby dosažení požadavků na regulaci inverzním modelem [2].

$$w_R(t) = \frac{B(z^{-1})}{A(z^{-1})} \tag{3.26}$$

Důvodem této úpravy, také často bývá, že neuronová síť jinak reguluje příliš rychle, kde důsledkem je obrovský akční zásah. Vhodnou volbou referenčního modelu, který zpomalí odezvu inverzního modelu se dá dosáhnout zásadního zmenšení akčního zásahu i požadavků na regulaci. Kritérium k minimalizaci v případě specializovaného trénování je v podobě rovnice (3.28).

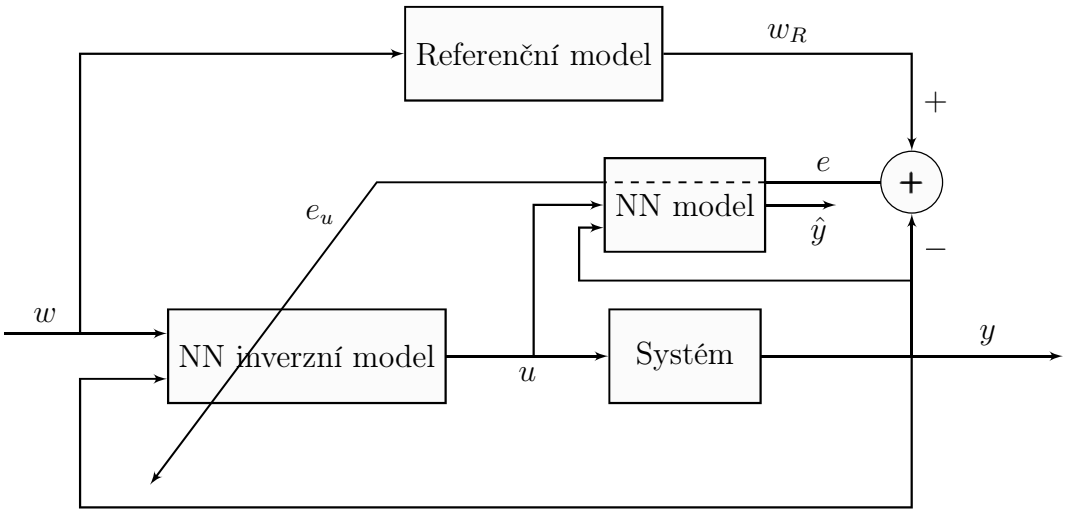
$$J = \frac{1}{2N}\sum_{t=1}^{N}[w_R(t) - \hat{y}(t)]^2 \tag{3.27}$$

Kde w_R symbolizuje žádanou veličinu w, která prošla přes referenční model. Velkou ne-

výhodou je že tato metoda je poměrně složitá a výpočetně náročná, protože je potřeba využít modelu systému k výpočtu backpropagation. Protože je výstup modelu systému závislý na čase, je třeba inverzi k systému získat metodou pro učení rekurentních sítí, nebo vhodně aproximovat gradient.Gradient je tedy ve tvaru závislém na čase.

$$\frac{dy(t)}{dW} = \frac{\partial y(t)}{\partial u(t-1)} \frac{du(t-1)}{dW} \tag{3.28}$$

První člen této rovnice je jakobián soustavy, který nemusí být znám. Odhad jakobiánu soustavy, při použití této metody lze získat právě naučením neuronové sítě jako modelu soustavy. Druhý člen je potom rekurzivní derivace vstupu modelu systému podle hledaných vah inverzního modelu.



Obr. 3.3. Specializované trénování

V první fázi specializovaného trénování, schématicky naznačeného na (Obr. 3.3), je naučena umělá neuronová síť s daty získanými ze systému. Je zvolen referenční model. Následuje učení inverzního modelu, při online učení je třeba chybu na výstupu $e = w_R - y$ backpropagovat skrz model systému tvořený umělou neuronovou sítí a použít ji k získání nových vah inverzního modelu. Popřípadě lze použít offline metodu, kdy se backpropaguje chyba $e = w_R - \hat{y}$.

3.2.3 Linearizace zpětné vazby

Linearizace zpětné vazby je jedna z metod, která se stala velmi oblíbenou. Rozdílem oproti předchozím metodám je, že poskytuje velmi dobré výsledky, bez nutnosti použití rekurentní neuronové sítě. Důvodem je použití modelu který aproximuje běžný NNARX. Aproximace je provedena, podle rovnice (3.29), vyjmutím posledního známého akčního zásahu z nelinearity a rozdělením systému na dvě funkce. Vyjmutý akční zásah je vhodné nahradit akčním zásahem v budoucím kroku, protože se tím zjednoduší výpočet, podle [9].

$$\begin{aligned}\hat{y}(t) = f[y(k-1),\ldots,y(k-n),u(k-2)\ldots,u(k-m)] \\ + g[y(k-1),\ldots,y(k-n),u(k-2)\ldots,u(k-m)]u(k)\end{aligned} \tag{3.29}$$

Funkce aproximující vstup je označena jako g a funkce aproximující výstup je označena f. Linearizace zpětné vazby, pro získání hodnoty akčního zásahu je potom provedena nahrazením hodnoty výstupu hodnotou žádané veličiny podle rovnice (3.30).

$$u(k) = \frac{w(k) - f(k)}{g(k)} \tag{3.30}$$

Tato metoda byla navržena pro linearizaci a následný návrh zpětnovazebního regulátoru ve vnější smyčce. Nevýhodou této metody je, že systém musí být ve všech oblastech stabilní.

3.2.4 Přímo-vazební regulace

Přímo-vazební regulace je metoda vylepšující průběh regulace využívající stávajících regulátorů. V některých případech v praxi může nastat problém že nelineární systém byl aproximován lineárním a pro něho byl navrhnut regulátor. Po určité době se potom přijde na to, že systém je až příliš nelineární a nelinearity jsou špatně popsatelné. Pokud je tento regulátor implementován softwarově potom ho lze jednoduše vylepšit touto metodou. Tato metoda využívá inverzní metody systému s připojeným regulátorem akční zásah je tedy tvořen přímo-vazební složkou a složkou zpětnovazební (3.31), kde složka přímo-vazební je cílem učení inverzního modelu [2].

$$u(k) = u(k)_{feedforward} + u(k)_{feedback} \tag{3.31}$$

Problémem této metody že pokud není znám model tak určení doplňkového přímovazebního zásahu není jednoduché a je třeba získat model systému např. identifikací umělou neuronovou sítí. V případě, pokud je model znám, tak se úloha značně zjednodušuje pouze jednoduchým výpočtem. Dále je možné tento regulátor úplně nahradit tabulkou hodnot a to v případě, kdy se regulační pochod opakuje.

3.2.5 Metoda okamžité linearizace

Linearizace je jedna z běžných technik, kterou lze navrhnout řízení nelineárních systémů. Jejím principem je rozvoj nelineárního popisu do prvních dvou členů taylorovy řady, okolo pracovního bodu. Důvodem velké oblíbenosti této metody je možnost aplikace širokých a většinou jednoduchých technik návrhu lineárních regulátorů. Lineární reprezentace systému může být dostatečná ovšem většinou to tak není. Lepší možností by bylo linearizovat nelineární systém v několika bodech a měnit modely v závislosti na daném pracovním pásmu. Potom pro každou pracovní oblast navrhnout regulátor. Tato metoda je ovšem poměrně složitá protože pro každou pracovní oblast je nutné navrhnou regulátor.

Pokud jsou ale k dispozici data získaná na nelineárním systému, tak potom lze naučit

dopřednou umělou neuronovou síť a posléze ji linearizovat. Ovšem zásadní rozdíl v linearizaci umělé neuronové sítě oproti standardní linearizaci modelu systému je, že linearizace bude provedena ve všech bodech, kde byly vzorky pro učení sítě. Jedná se tedy o jakýsi obrácený postup, než při běžném návrhu nelineárního regulátoru, kdy je systém nejdříve linearizován a potom diskretizován [2]. Pokud je tedy známý nelineární model v podobě umělé neuronové sítě $\hat{y}(k) = g[\Theta(k)]$ s regresorem $\Theta(k) = [y(k-1), \ldots, y(k-n), u(k-d) \ldots, u(k-d-m)]^T$ potom derivací modelu podle jednotlivých prvků regresního vektoru v každém vzorku, lze získat koeficienty lineární diferenční rovnice (3.32) pro daný vzorek umělé neuronové sítě.

$$\hat{y}(k) = -a_1\hat{y}(k-1) - \ldots - a_n\hat{y}(k-n) + b_0u(k-d) + \ldots + b_nu(k-d-m). \quad (3.32)$$

Tato metoda se velmi podobá adaptivnímu řízení, kdy je model lineární model získán v každém kroku. Tato metoda má tedy stejné problémy jako adaptivní řízení, tedy pokud dochází ke změně operační oblasti velmi rychle systém nemusí jít regulovat [2].

3.3 Prediktivní regulátory s neuronovou sítí

Prediktivní řízení je jednou z oblíbených metod návrhu regulátoru. Prediktivní řízení je nejčastěji založeno na diskrétním modelu. Existuje několik podmínek, které je nutné splnit pro prediktivní řízení [4].

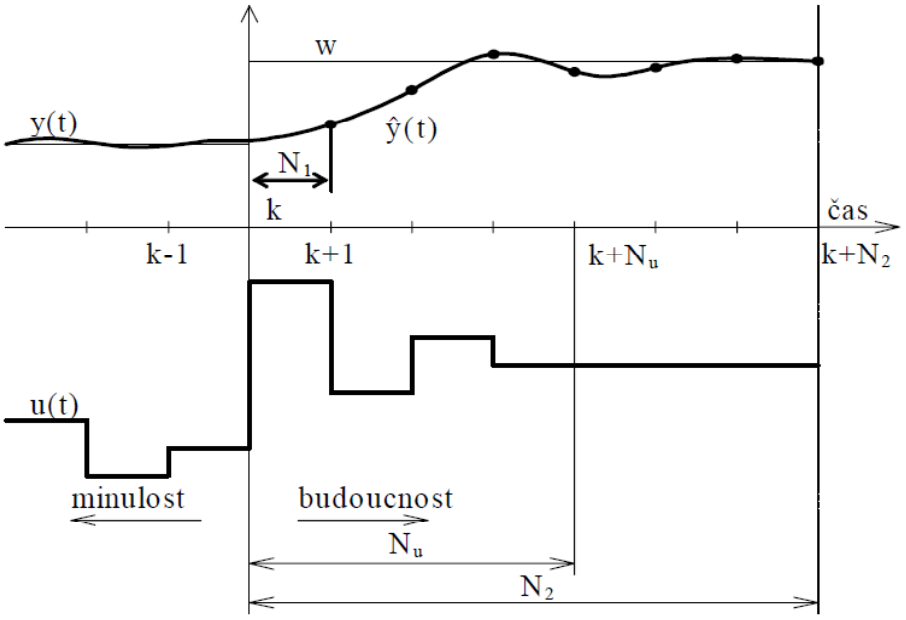
- Je třeba znát budoucí žádanou veličinu.
- Matematický model systému musí být použitelný k predikci.

Pokud jsou tyto podmínky splněny, tak regulace probíhá tak, že je určen počet následujících kroků, které budou vystupovat v kritérii pro minimalizaci. Kritérium se minimalizuje v každém okamžiku kdy se ve zvoleném horizontu mění žádaná veličina. Minimalizované kritérium má rovnici (3.33).

$$F = \sum_{i=N_1}^{N_2} \delta(i) \left[\hat{y}(k+i) - w(k+i)\right]^2 + \sum_{i=1}^{N_u} \lambda(i) \left[\Delta u(k+i-1)\right]^2 \quad (3.33)$$

Kde $\hat{y}$ je odhad výstupu získaný pomocí modelu, w žádaná veličina, Δu je přírůstek akčního zásahu, N_1 je minimální horizont, N_2 je maximální horizont a N_u je horizont řízení.

Toto kritérium má velkou výhodu v tom, že lze omezit jak akční zásah, tak regulovanou veličinu. Metody prediktivního řízení lze snadno aplikovat na téměř veškeré typy systémů.



Obr. 3.4. Princip prediktivního řízení [10]

Otázkou je tedy, kde využit neuronovou síť v prediktivním řízení. Už několikrát bylo zmíněno, že prediktivní řízení využívá modelu a zde lze použít neuronovou síť. V případě prediktivního řízení, jde vždy o rekurentní neuronové sítě, které lze s výhodou použít pro predikcí velkého počtu kroků dopředu [1].

Nevýhodou je, že nelineární prediktivní řízení, při použití umělé neuronové sítě, je výpočetně velmi složité. Důvodem je, že v každém kroku je potřeba provádět optimalizaci akčních zásahů. Proto lze s výhodou využít již popsané metody okamžité linearizace [2].

4 Regulátory použité k porovnání

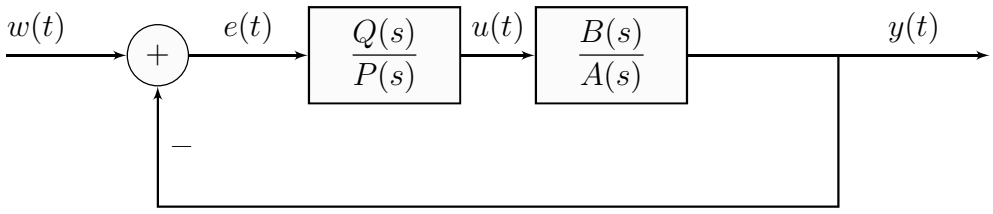
K porovnání regulátoru založených na neuronových sítích byly vybrány algoritmy, které by mohly být zvoleny v jiném případě, kdy není známý model. V úvahu připadají různé metody identifikace a průběžné identifikace. Jako první porovnávaný je jednoduchý regulátor vytvořený na základě lokální identifikace a jako druhý algoritmus je samočinně se nastavující regulátor.

4.1 Umístění pólů v operační oblasti

Tato metoda je založena na identifikaci v operační oblasti. Pro identifikaci bylo rozhodnuto, že nejvhodnější metodou je identifikace z přechodové charakteristiky. Důvodem je, že komplikovanější metody u nelineárních systémů nepřinesou výrazně lepší výsledky. Zároveň je jednoduše možné touto metodou identifikovat systém v operačním bodě. Identifikace přechodové charakteristiky bude prováděna automaticky metodou Nelder-Mead simplex.

4.1.1 Umístění pólů

Umístění pólů je velmi jednoduché a současně je to velmi silná metoda návrhu lineárního regulátoru, která dává kontrolu nad výsledným regulačním pochodem.Pokud je regulační obvod ve standardním tvaru, jako na obrázku (Obr. 4.1).



Obr. 4.1. Standardní struktura regulačního obvodu

Potom přenos řízení je získán v podobě rovnice (4.1).

$$G_{w/y} = \frac{Y(s)}{W(s)} = \frac{\frac{Q(s)}{P(s)}\frac{B(s)}{A(s)}}{1 + \frac{Q(s)}{P(s)}\frac{B(s)}{A(s)}} = \frac{Q(s)B(s)}{A(s)P(s) + Q(s)B(s)} \tag{4.1}$$

Následuje přiřazení pólů v polynomu $D(s)$ přenosu řízení v podobě řešení diofantické rovnice (4.2).

$$A(s)P(s) + Q(s)B(s) = D(s) \tag{4.2}$$

Řešením rovnice (4.1) se získá rovnice regulátoru (4.3).

$$G_R(s) = \frac{Q(s)}{P(s)} \tag{4.3}$$

Pro získání regulátoru v diskrétním tvaru lze odvodit podobný postup, nebo použít

Tustinovu transformaci (pro zachování spektra). Tuto metodu lze použít k přiřazení pólů do téměř jakékoliv řídící struktury např. 2DOF a jiné.

4.2 Adaptivní regulátory

Adaptivní systém má tři vstupy a jeden výstup. Na adaptivní systém působí žádaná veličina w a porucha n. Dalším vstupem je informace o požadovaném chování systému Ω. Ve většině případů porucha není známa, žádaná veličina a chování systému jsou známé. Toto lze popsat pravidlem (4.4).

$$y = f(w, n, \Phi) \tag{4.4}$$

Toto pravidlo přiřazuje každé kombinaci vstupů jedno Φ a právě takové aby bylo minimem Φ^* funkce g[4].

$$g(\Omega, w, v, \Phi^*) = \min g(\Omega, w, n, \Phi) \tag{4.5}$$

Adaptace je tedy proces při kterém se hledá optimální parametry Φ^* dokud nejsou nalezeny, potom v závislosti na systému může být adaptace vypnuta.

4.2.1 Samočinně se nastavující regulátory

Samočinně se nastavující regulátory jsou podskupinou adaptivních regulátorů, která často odděluje řídící a identifikační část. Tohoto zjednodušení je dosaženo tzv. vnucenou separací identifikace a řízení. Princip spočívá v následujících zjednodušení.

- Vektor parametrů modelu je v daném kroku považován za známý. A to předpokladem že minulý odhad bude podobný jako současný.
- Na základě algoritmu řízení je potom možné v každém kroku vypočítat $u(k)$.
- Získáním nového vzorku $y(t)$ a známého akčního zásahu $u(k)$ se provede další identifikační krok, pomocí rekurzivní identifikace. Nová informace je tedy použita k updatování minulých odhadů parametrů.

To že se struktura řízení a identifikace rozpadá je velmi výhodné. Umožňuje to velmi rychlou a jednoduchou změnu řídícho algoritmu. Také to snižuje možnost chyby protože jediné spojení identifikace a řízení je odhad vektoru parametrů soustavy.

5 Kritéria pro porovnání regulací

Pro porovnání regulací bylo vybráno několik kritérií a to jak typických pro regulaci, tak i pohledy z jiných hledisek.

5.1 Pohled z regulačního hlediska

Pro vyhodnocení kvality regulace z regulačního hlediska byla vybrána rovnice (5.1) sumace kvadrátů regulační odchylky $e(k) = w(k) - y(k)$.

$$S_y = \frac{1}{k_2 - k_1 + 1} \sum_{k=k_1}^{k_2} e^2(k) \tag{5.1}$$

Důvodem použití tohoto kritéria je, že na jeho základě lze jakýmsi způsobem posoudit, jak je rychlá regulace. Protože čím je menší suma kvadrátů regulačních odchylek tím rychleji došlo k dosažení žádané hodnoty. Dalším kritériem je sumace přírůstků akční veličiny (5.2).

$$S_u = \frac{1}{k_2 - k_1 + 1} \sum_{k=k_1}^{k_2} \Delta u^2(k) \tag{5.2}$$

toto kritérium sleduje jak moc se mění akční veličina. Akční veličina je často v reálných systémech silně omezená a proto je toto důležitý ukazatel.[4] Vhodný ukazatel regulace je, také relativní překmit, který udává jak moc překmitla regulovaná veličina žádanou. Tento ukazatel může být zásadní, protože překmit u reálného systému může mít za následek vážné poškození. Posledními ukazateli regulace, které byly zvoleny jsou $\min u(k)$, $\max u(k)$, $\min y(k)$ a $\max y(k)$. Tyto ukazatele ukazují extrémy regulačního pochodu a mohou tedy být užitečné.

5.2 Pohled z jiných hledisek

Regulační hlediska nejsou jediná z pohledu reálné aplikace, proto byly zvoleny další kritéria. Při implementaci algoritmu existuje několik možných problémů, jedním z nich je potřebný počet výpočtů v každém kroku regulace. Proto jedním z porovnávaných parametrů bude taky počet operací násobení a sčítání. Dalším možným problémem je paměťové omezení v reálných aplikací, které bude také zkoumáno jako další kritérium.

6 MATLAB

MATLAB je matematický software, který k řešení úloh využívá příkazové řádky a skriptů ve skriptovacím jazyce MATLAB. Tetnto jazyk byl původně jen matematického charakteru ale postupně se rozrostl tak že je možné ho použít k programování jednoduchých aplikací. Již dlouhou dobu MATLAB patří ke stálicím ve vědě, výzkumu a průmyslu, při řešení různých matematických problémů[8]. MATLAB v základu nabízí velké množství příkazů, které jsou dále rozšiřitelné přídavnými toolboxy. Nejčastější úkoly v rámci MATLAB je práce s maticemi, vykreslování grafů, simulace a analýza, nebo vytvoření aplikace s uživatelským rozhraním.

Důležitou součástí MATLAB se stal také Simulik, který umožňuje dataflow programování, se zaměřením na simulace. Simulink nabízí velké množství různých řešitelů a nastavení. V základu je vhodný převážně na spojité a diskrétní simulace, ale po rozšíření tzv. blocksety umožňuje fyzikální simulace, simulace DSP, HDL, simulace částí vozidel a letadel, nebo právě simulace neuronových sítí. Další velkou výhodou Simuliku je, že z vytvořeného schématu dokáže vygenerovat kód v C/C++ nebo generaci kódu ve Verilog pro sysntézu HDL, popřípadě umožňuje schéma převést na analogové zapojení elektrických součástek.

6.1 Neural network tooolbox

Neural network toolbox je rozšíření MATLAB, které umožňuje jednoduchou práci s neuronovými sítěmi jak v prostředí MATLAB tak v Simulink. Poskytuje nástroje pro definici podoby modelu neuronové sítě, její učení a následnou vizualizaci výsledků. Dále umožňuje řešit klasické problémy strojového učení jako klasifikace, regrese, clustering, redukci dimenze dat, předpověď datové série modelovaní a řízení dynamických systémů. Toolbox také obsahuje část zaměřenou na klasifikaci obrazu s využitím konvolučních umělých neuronových sítí. Dále je možná spolupráce s Paralel Computing Toolbox, která umožňuje převedení kódu na instrukce pro řešní na GPU, nebo využít Matlab Distributed Computing Server pro velmi složitých sítí [7].

6.1.1 Regulátory dostupné v Neural Network Toolbox

V Neural Network Toolboxu je implementováno několik regulátorů, které lze přímo použít. Prvním je nazvaný Model-Reference a jedná se o regulátor v podobě inverzního modelu, který je trénován ve speciálním uspořádání. Další dostupnou možností v je regulátor označený jako NARMA-L2. Jedná se o stejný přístup jako v kapitole o linearizaci zpětné vazby. Tento regulátor je trénován specializovanou metodou. Posledním dostupným regulátorem je nelineární prediktivní regulátor. Tento regulátor pracuje stejně jak je popsáno v kapitole o prediktivních regulátorech.

II. PRAKTICKÁ ČÁST

7 Implementace přímého inverzního řízení

Při implementaci algoritmů neuronových sítí je třeba dávat pozor na možné chyby, proto navržený postup začíná volbou systému, na kterém jsou prováděny testy. Následně je v této kapitole popsána implementace přímého inverzního řízení a porovnání regulace s jinými typy regulátorů v simulaci. Porovnání regulátorů je provedeno, jak podle integrálních regulačních kritérií, tak podle potřeby paměti tak podle potřebných matematických operací. Veškeré výsledky jsou uvedeny v tabulkové formě a regulační pochody jsou uvedeny v příloze. Po otestování implementace na benchmark systému se další kapitola bude zabývat ukázkou implementace na reálný systém.

7.1 Benchmark systém v simulaci

Nejdříve byl zvolen systém, na kterém bude prováděno testování algoritmů. Byl zvolen nelineární systém prvního řádu a to nádrž. Nádrž je typický systém s poměrně silnou nelinearitou v části statické charakteristiky. Nádrž byla popsána zjednodušeným modelem (7.1).

$$\frac{dh(t)}{dt} = \frac{q(t)Q_{in}}{S} - \frac{Q_{out}}{S}\sqrt{h(t)} \tag{7.1}$$

Kde Q_{in} představuje konstantu spojenou s přítokem, Q_{out} konstantu spojenou s odtokem, $h(t)$ je hladina a $q(t)$ je přítok. Model byl převzat z [16] a [17].

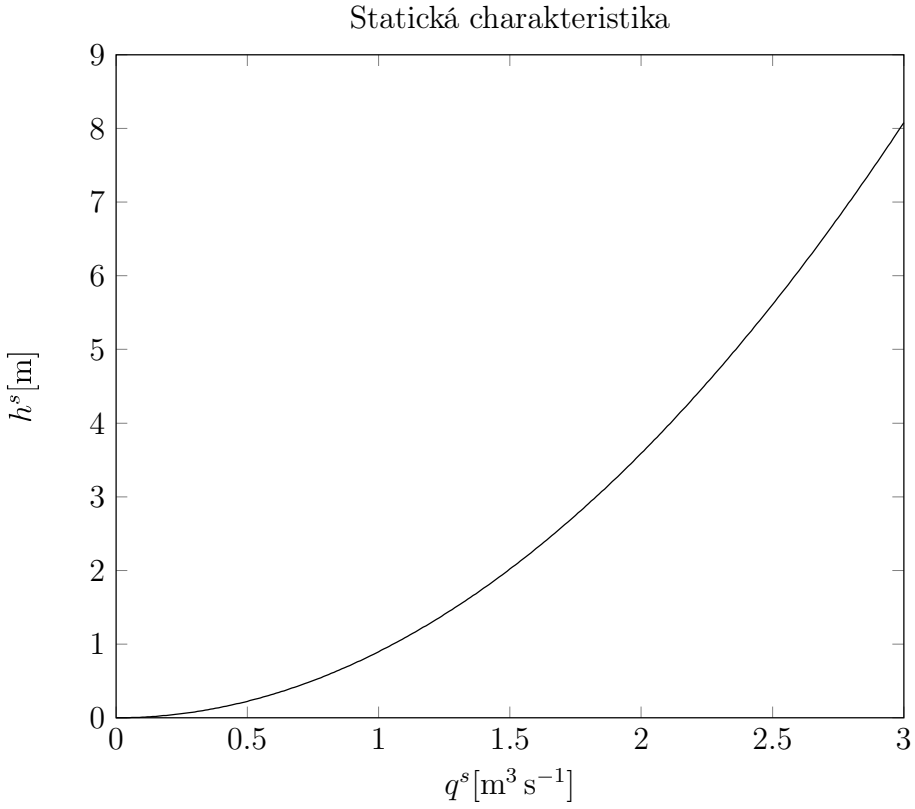
Tento model bude použit k porovnání mezi jednotlivými typy regulátorů. Následně po ověření funkce na tomto testovacím modelu, bude regulátor s umělou neuronovou sítí aplikován na reálný model podobné nádrže s jinými parametry.

7.1.1 Statická charakteristika

Prvním krokem, při průzkumu systému, je průzkum nelinearit. K tomuto účelu je vhodné nalézt statickou charakteristiku, obzvlášť pokud není znám model systému. V tomto případě je znám matematický model, pak stačí vyjádřit rovnici (7.1) pro $\frac{dh(t)}{dt} = 0$. Řešením lze získat rovnici (7.2).

$$h^s = \left(\frac{q^s Q_{in}}{Q_{out}}\right)^2 \tag{7.2}$$

Vynesením do grafu (Obr. 7.1) lze získat závislost ustálených hodnot přítoku na hladině.

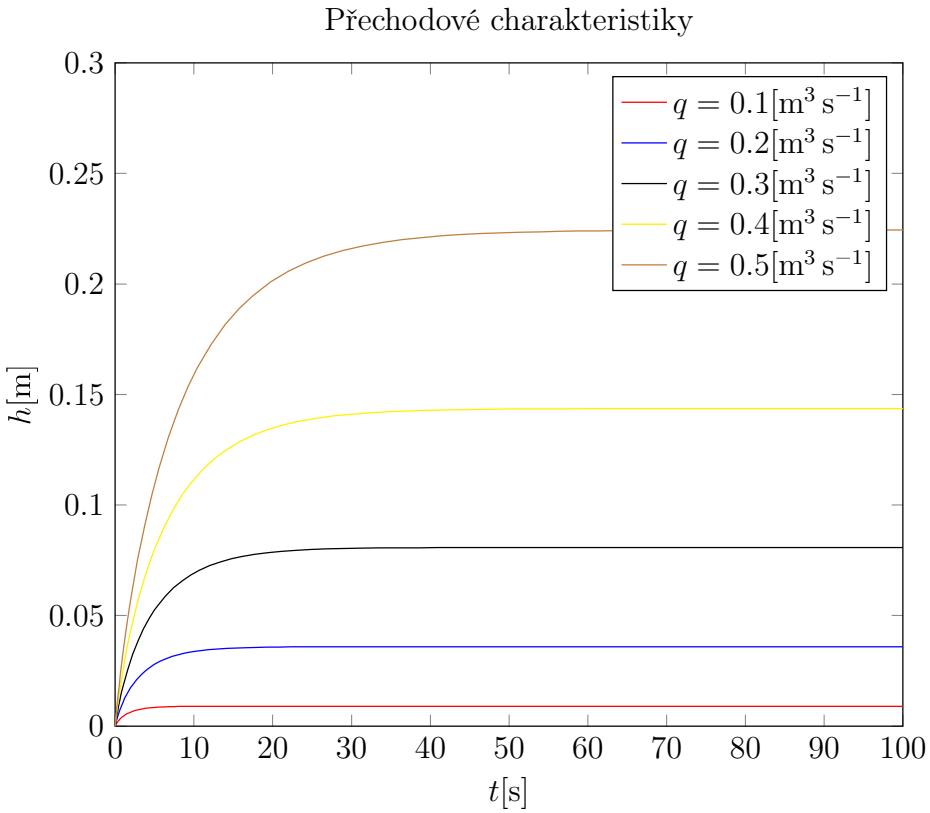


Obr. 7.1. Statická charakteristika benchmark systému

7.1.2 Přechodová charakteristika

Studiem přechodové charakteristiky v několika oblastech nelineárního systému lze získat dostatečné znalosti o dynamických jevech a vhodně zvolit vzorkovací frekvenci.

Z přechodové charakteristiky ve zvolených oblastech je patrné, že se poměrně dost mění časová konstanta a proto volba vzorkovací frekvence nebude úplně jednoduchá. Pracovní oblast, byla zvolena mezi $h =< 0.1, 1 > [m]$ a protože pro větší přítoky časová konstanta zvyšuje byla vzorkovací frekvence zvolena pro malý přítok. Ovšem je třeba dát si velký pozor, protože čím menší bude vzorkovací frekvence, tím víc to ovlivní diskrétní popis systému. Póly půjdou směrem k okraji jednotkové kružnice a nuly půjdou k nule a to může vést k numerické nestabilitě. Pokud je znám model systému jeho linearizací ve více bodech lze získat průběh pólů a nul (Obr. 7.2). V případě tohoto modelu se jedná o systém prvního řádu, takže bude mít pouze jeden pól.



Obr. 7.2. Přechodové charakteristiky pro různě zvolené přítoky

Z (Obr. 7.3) je patrné, že se póly blíží k hranici jednotkové kružnice. A zároveň je z přechodových charakteristiky patrné, že časová konstanta se velmi rychle zvětšuje s přítokem. Vzorkovací frekvence byla tedy zvolena 2s ohledem na pracovní rozsah.

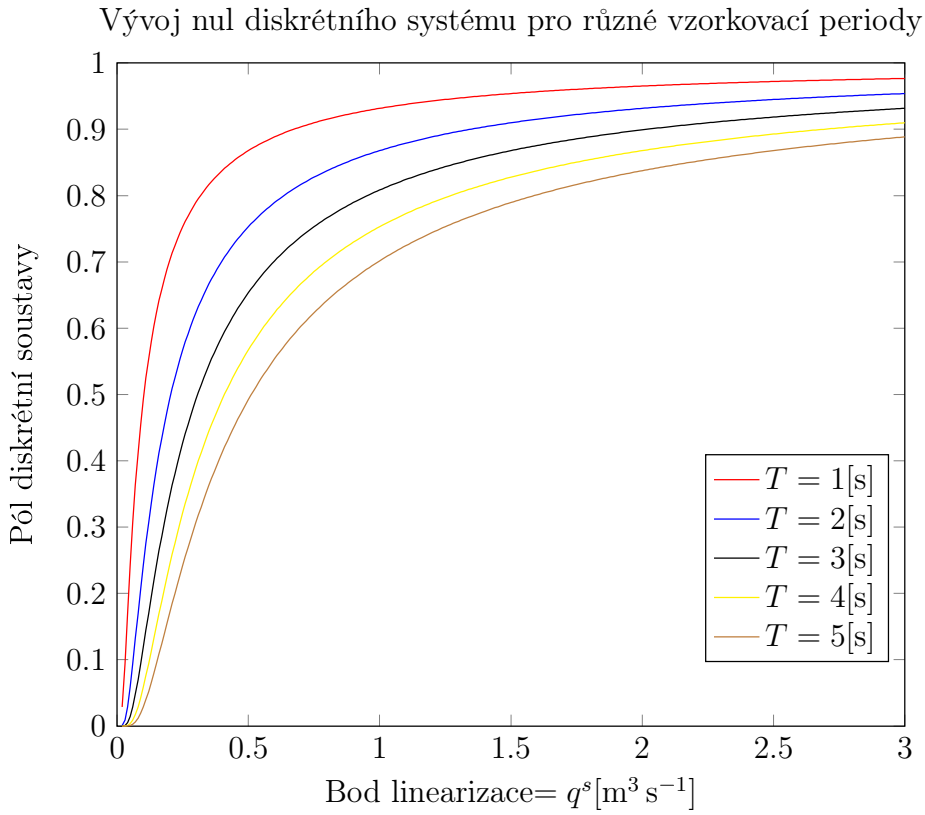
7.2 Implementace backpropagation

Nejdříve bylo rozhodnuto jak hluboká síť bude zvolena a jaký typ modelu. Model byl zvolen jako NNARX z důvodu jeho všestrannosti. Hloubka sítě byla stanovena jako běžná neuronová síť, tedy jen 3 vrstvy. První vrstva je vstupní vrstva, následuje jedna skrytá a jedna výstupní.

Aktivační funkce skryté vrstvy je nelineární a je to **tansig**(x), aktivační funkce výstupní vrstvy je lineární **identity**(x). Volba nelineární funkce je zřejmá, důvodem je, že řešené problémy budou nelineární. Výstupní aktivační funkce je lineární, z důvodu, že se jedná o kontinuální predikci hodnot. Kritérium bylo zvoleno jako kvadrát chyby.

Následně byly určeny derivace a bylo dosazeno do předpisů pro backpropagation, kde pro výstupní vrstvu byla nalezena citlivost podle rovnice (1.23). Byla zjištěna derivace aktivační funkce výstupu (7.4) a derivace kritéria (7.4).

$$\frac{\partial \boldsymbol{f}^2(\boldsymbol{n}^2)}{\partial \boldsymbol{n}^2} = 1 \tag{7.3}$$



Obr. 7.3. Vývoj nul diskrétního systému pro různé vzorkovací periody

$$\frac{\partial \boldsymbol{F}(\boldsymbol{a}^2)}{\partial \boldsymbol{n}^2} = \hat{\boldsymbol{y}} - \hat{\boldsymbol{t}} \tag{7.4}$$

Potom citlivost pro tuto vrstvu je vyjádřena jako (7.5).

$$\boldsymbol{s}^1 = \left(\hat{\boldsymbol{y}} - \hat{\boldsymbol{t}}\right) \tag{7.5}$$

Gradienty pro výstupní vrstvu lze nalézt pomocí rovnic (7.6) a (7.7).

$$\frac{\partial \boldsymbol{F}(\boldsymbol{a}^2)}{\partial \boldsymbol{W}^2} = \boldsymbol{s}^1 \left(\frac{2}{1+e^{-2\boldsymbol{n}^1}} - 1\right)^T \tag{7.6}$$

$$\frac{\partial \boldsymbol{F}(\boldsymbol{a}^2)}{\partial \boldsymbol{b}^2} = \boldsymbol{s}^1 \vec{\boldsymbol{1}}^T \tag{7.7}$$

Pro skrytou vrstvu je potom potřeba najít derivaci aktivační funkce (7.8) a derivaci pro citlivost (7.9).

$$\frac{\partial \boldsymbol{f}^1(\boldsymbol{n}^1)}{\partial \boldsymbol{n}^1} = 1 - \left(\frac{2}{1+e^{-2\boldsymbol{n}^1}} - 1\right)^2 \tag{7.8}$$

$$\boldsymbol{s}^2 = \left(1 - \left(\frac{2}{1+e^{-2\boldsymbol{n}^1}} - 1\right)^2\right) \left(\boldsymbol{W}^2\right)^T \boldsymbol{s}^1 \tag{7.9}$$

Gradienty pro skrytou vrstvu jsou potom určeny rovnicemi (7.10) a (7.11).

$$\frac{\partial \boldsymbol{F}(\boldsymbol{a}^1)}{\partial \boldsymbol{W}^1} = \boldsymbol{s}^2 \boldsymbol{x}^T \tag{7.10}$$

$$\frac{\partial \boldsymbol{F}(\boldsymbol{a}^1)}{\partial \boldsymbol{b}^1} = \boldsymbol{s}^2 \vec{\boldsymbol{1}}^T \tag{7.11}$$

Nejdřív byl vytvořen kód přímého průchodu neuronovou sítí (Prog. 7.1). Teto kód je

upraven tak, aby váhy a prahy byly v jedné matici. Důvodem je značné zrychlení řešení kódu matlabem, který umí využívat vektorové operace moderních procesorů.

Prog. 7.1. Dopředný průchod

```
function [Z1,A1,A2]=feed_forward_matrix(W1,W2,sample)
TanSig=@(x)(2./(1+exp(-2.*x))-1);
Z1 = W1'*sample;
A1=TanSig(Z1);
Z2 = W2'*[A1;ones([1,length(A1)])];
A2=Z2;
```

Samotná backpropagation je provedena podobným způsobem (Prog. 7.2). Nejdříve je vypočítán přímý průchod a potom jsou vypočítany citlivosti a následně jsou předány funkci, která se stará o výpočet nových vah.

Prog. 7.2. Postup nalezení gradientů

```
s1 = A2 - train_target;
error=sum(s1.^2);
s2 = (1-tansig(Z1).^2).*(w2(1:hidden,:)*s1);
%gradienty
W1_gradient=train_samp*s2';
W2_gradient=[TanSig(Z1);ones([1,length(TanSig(Z1))])]*s1';
```

Výpočet nových vah je prováděn jedním ze dvou použitých algoritmů a to buď adaptivní změnou délky gradientu, nebo standardní sestup gradientu. Program (Prog. 7.3) umožňuje i nastavení momentu, pro vhodné metody výpočtu gradientu.

Prog. 7.3. Standardní trénování s možností členu momentu

```
function [w1,w2,w1UPDATE,w2UPDATE]=update_W_momentum(...
w1,w2,W1_grad,W2_grad, ...
W1up_last,W2up_last,learning_rate,momentum)
w1UPDATE=W1up_last*momentum-learning_rate*W1_grad;
w2UPDATE=W2up_last*momentum-learning_rate*W2_grad;
w1 = w1 + w1UPDATE;
w2 = w2 + w2UPDATE;
```

Ukončení trénování bylo na základě validační množiny. Tedy umělá neuronová síť byla trénována podle trénovací množiny, ale o ukončení se rozhodlo podle trendu chyby na validační množině. Jedná se o typ ukončení, který zlepšuje generalizaci dat. V případě simulace je ovšem dostatek dat na malý počet neuronů, takže chyba na validační množině klesá s chybou na trénovací množině.

7.2.1 Implementace generování a přípravy dat

Generování a příprava dat, pro umělou neuronovou síť není lehkým úkolem, obzvlášť pokud se jedná o regresi dynamických jevů. Výsledky budou záviset jak periodě vzorkování tak na pokrytí různých frekvencí. Jako signál byl využit „skyline“, kde je volitelným parametrem pravděpodobnost, která určuje jestli dojde ke změně nebo ne. Ukázka

tohoto signálu je například v grafu (Obr. 7.5). Generování je prováděno, podle již popsaných vzorců, pomocí vytvořené MATLAB funkce (Prog. 7.4). Parametrem funkce je také seed pro generátor náhodných čísel, aby bylo možné opakovat generování dat se stejným výsledkem.

Prog. 7.4. Generování signálu pro identifikaci

```
function [data,time]=gen_sig(samples,alfa,Ts,var,mean,clip,
   seed)
data=[0];
rng(seed);
for i=1:samples
    if rand(1)>alfa
     data=[data randn(1)*var+mean];
    else
     data=[data data(end)];
    end
end
```

Výsledná data je ještě potřeba omezit z důvodu, že budou působit jako akční zásah na soustavu a pokud zásah nebude omezen, mohl by reálnou soustavu poškodit. Což je provedeno programem (Prog. 7.5).

Prog. 7.5. Omezení dat

```
if ~isempty(clip)
    low=clip(1);
    up=clip(2);
    data(data>up)=up;
    data(data<low)=low;
end
```

Funkce také provede vygenerování času k zadaným vzorkům podle vzorkovací periody a vypočítá spektrální hustotu neořezaného signálu, která může být použita pro odhad zda je identifikační signál vhodný pro danou soustavu.

Následuje normalizace dat do rozsahu ± 1. Funkce (Prog. 7.6) vrací pole hodnot, které umožní normalizovat nové data i převést normalizované data zpět na skutečné hodnoty.

Prog. 7.6. Normalizace dat do rozsahu ± 1

```
function [scaled,rescale]=scaler(series,rescale,re)
if length(rescale)==1
    rescale=[min(series),max(series)];
    scaled=2*(series-min(series))./(max(series)-min(series)
       )-1;
else
    if re==1
    MIN=rescale(1);
    MAX=rescale(2);
    scaled=(series+1).*(MAX-MIN)/2+MIN;
    else
    MIN=rescale(1);
    MAX=rescale(2);
```

```
    series(series>MAX)=MAX;
    series(series<MIN)=MIN;
    scaled=2*(series-MIN)./(MAX - MIN)-1;
    end
end
```

Posledním krokem přípravy dat je rozdělení na trénovací, validační a testovací množinu. Je možné data náhodně rozdělit parametrem shuffle, nebo ponechat vzorky sekvenčně pro umělou neuronovou rekurentní síť.

Prog. 7.7. Rozdělení dat pro trénování

```
function [train_samp,train_target,test_samp,test_target,
   val_samp,val_target]=test_train_split(dsamp,dtarget,
   testtrain,testvalid,shuffle)
Nsamp=length(dtarget(1,:));
if shuffle==1
        shuff=randomperm(Nsamp);
        dsamp=dsamp(shuff);
        dtarget=dtarget(shuff)
end
train_set=round(Nsamp*testtrain);
test_set=round((Nsamp-train_set)*testvalid);
valid_set=Nsamp-train_set-test_set;
train_samp=dsamp(:,1:train_set);
train_target=dtarget(:,1:train_set);
test_samp=dsamp(:,(train_set+1):(test_set+train_set));
test_target=dtarget(:,(train_set+1):(test_set+train_set));
val_samp=dsamp(:,(test_set+train_set+1):end);
val_target=dtarget(:,(test_set+train_set+1):end);
fprintf('Samples: %d, Train: %d, Test: %d, Valid: %d \n',
   train_set+test_set+valid_set,train_set,test_set,
   valid_set);
```

Po rozdělení na jednotlivé množiny je zahájeno trénování neuronové sítě.

7.3 Ověření algoritmu na modelu

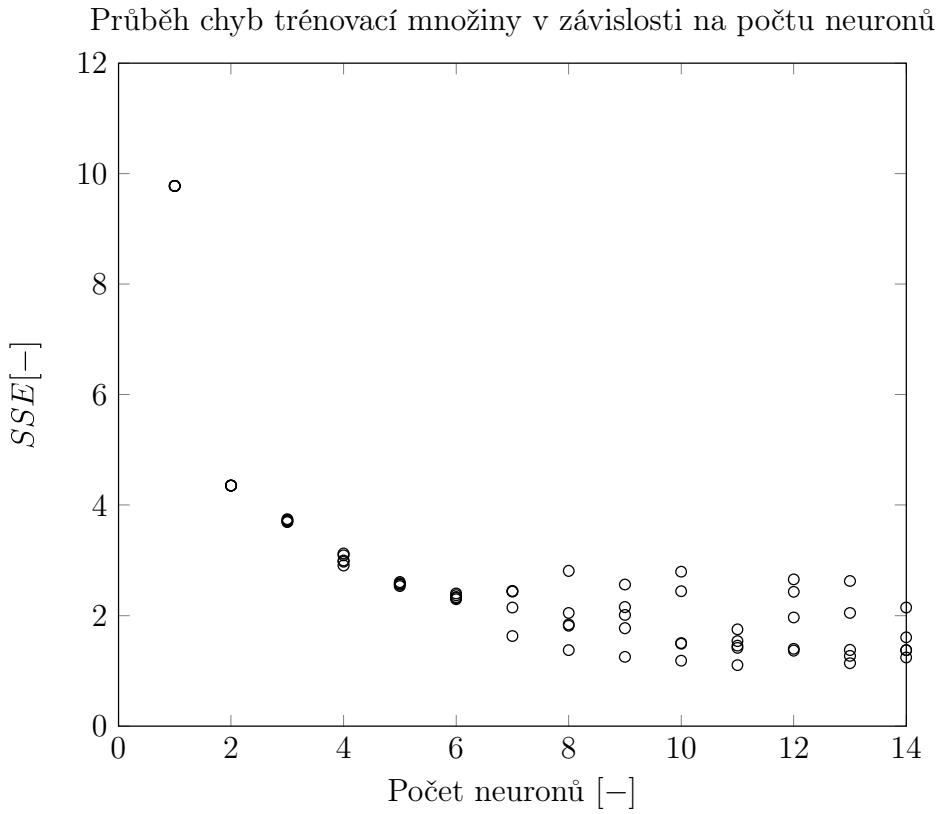
Po naprogramování veškerých nutných metod, nastal čas na testování. Při testování přímého modelu je třeba naleznout vhodnou délku vektorů zpožděných dat, pro učení umělé neuronové sítě. Vhodně vygenerovat data a připravit je pro neuronovou síť. Zvolit počet neuronů umělé neuronové sítě a natrénovat ji.

7.3.1 Generování a příprava dat

Pro generování a přípravu dat jsem již zvolili vzorkovací frekvenci z prvotních přechodových charakteristik na 2s. Data budou vygenerována pro několik α a výsledky budou porovnány na trénování neuronové sítě.

Volbu počtu neuronů jsem provedl poměrně jednoduchou metodou. Provedl jsem trénování neuronové sítě s různým počten neuronů. Učení pro každý případ bylo pro-

vedeno několikrát, ukončení učení bylo při delším rostoucím trendu validační množiny.



Obr. 7.4. Porovnání počtu neuronů

Z grafu (Obr. 7.4) je patrné, že přibližně do 6 neuronů je jedno jasné minimum ale přidáním dalších neuronů dojde k zvýšení počtu lokálních minim. Proto následující umělé neuronové sítě bylo použito 6 neuronů.

7.3.2 Přímý model

Účelem přímého modelu je otestování funkčnosti sítě. Tento model by také mohl být použitý jako prediktor o jeden krok dopředu. Důvodem je, že přímý model udává vztah mezi současnými vstupy a výstupem o jeden krok dopředu. Jedná se tedy o aproximaci funkce.

Vektor zpoždění byl určen jako dostatečný pro tři minulé hodnoty [15] [2]. Forma vektoru zpoždění je následující

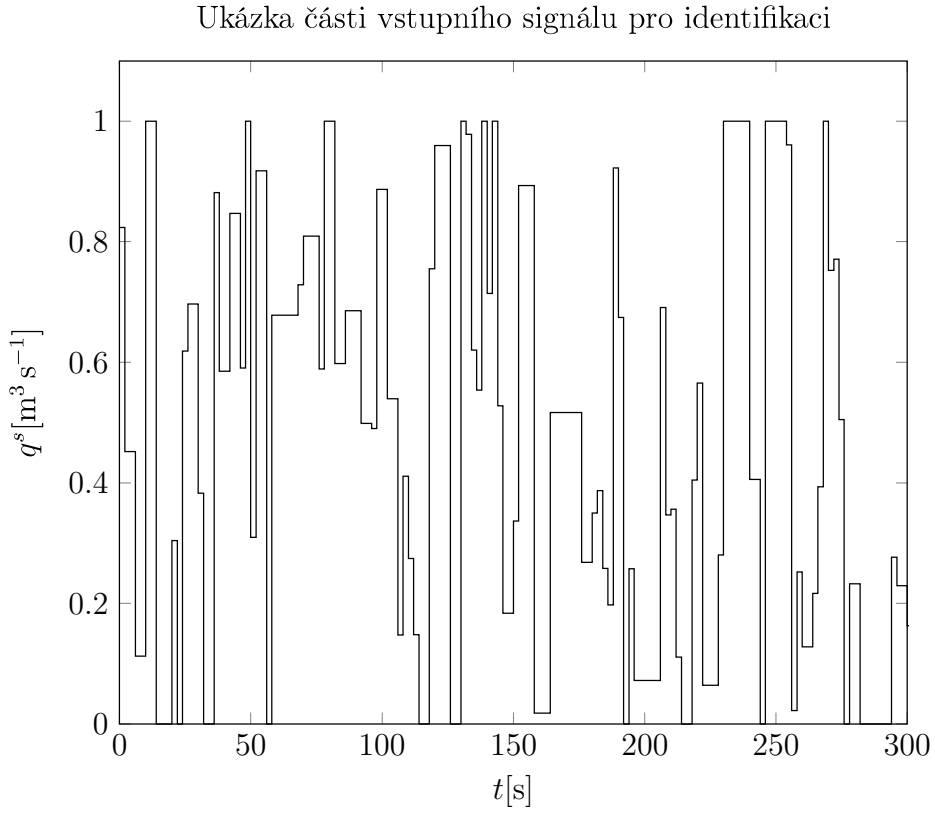
$$[y(k-3), y(k-2), y(k-1), u(k-3), u(k-2), u(k-1)] \tag{7.12}$$

Kde cílem učení je potom $y(k)$. Ve výsledku tedy posunutím času do sítě budou vkládat hodnoty v následující podobě.

$$[y(k-2), y(k-1), y(k), u(k-2), u(k-1), u(k)] \tag{7.13}$$

Výstupem umělé neuronové sítě je budoucí hodnota výstupu systému $y(k+1)$ Následně

byly vygenerovány data pro identifikaci modelu



Obr. 7.5. Ukázka identifikačního signálu

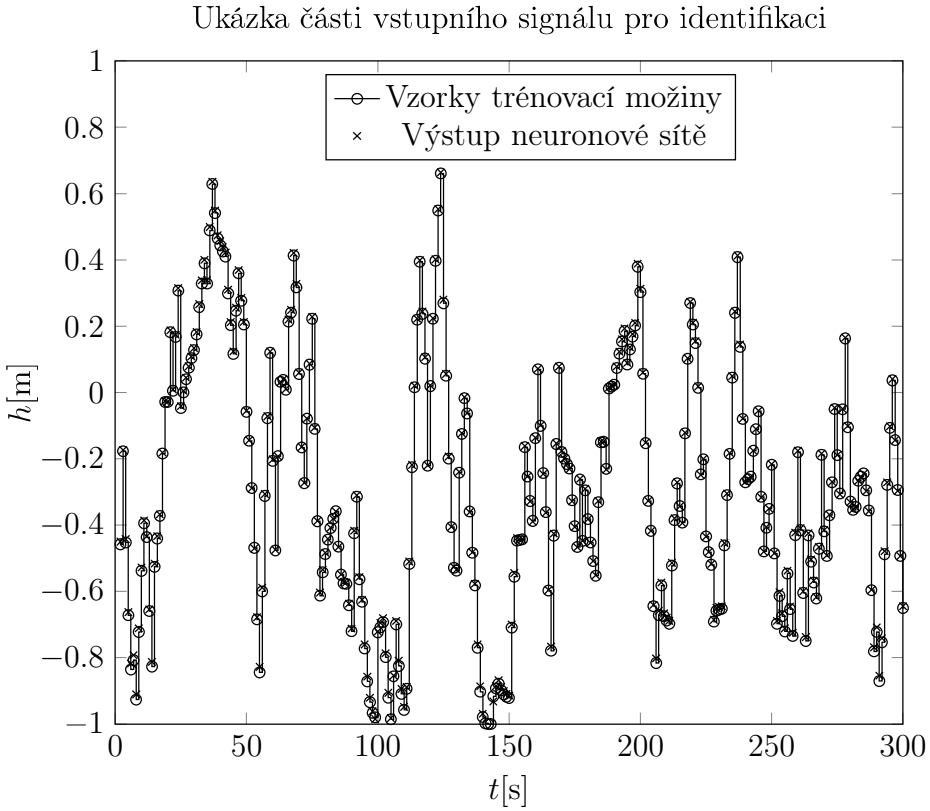
Tento signál byl použit k identifikaci systému a následně vytvoření dat pro učení umělé neuronové sítě.

Samotná umělá neuronové síť byla vytvořena v prostředí simulink z důvodu, že skutečný model v laboratoři komunikuje s modelem v simulinku. Model je na obrázku v příloze VII.. Nejdříve jsou data změněna do rozsahu ± 1 a následně použita jako vstup do sítě. Výsledek je převeden zpět do původního rozsahu zpožděn o jeden krok a porovnán se simulovaným výstupem nádrže.

Neuronová síť byla naučena na předem uvedených vstupech a jim odpovídajícím výstupech. Graf (Obr. 7.6) ukazuje porovnání mezi cílem učení a výstupem neuronové sítě. Je celkem dobře viditelné, že aproximace systému je ve většině oblastí velmi dobrá.

7.3.3 Inverzní model

Použití inverzního modelu je velice podobné. Nastává ovšem několik nutných úvah ohledně vektorů zpoždění, jak bylo popsáno v kapitole o přímém inverzním modelu. Nový vektor zpoždění musí obsahovat jednu hodnotu navíc, která bude nahrazena žádnou hodnotou a musí také vhodně upravit posunutí v čase u akčního zásahu (dopravním zpožděním). Vektor zpoždění je v tomto případě vyjádřen rovnicí (7.14).



Obr. 7.6. Ukázka porovnání části cílů a výstupů neuronové sítě

$$[y(k-1), y(k), y(k+1), u(k-1)] \tag{7.14}$$

Cílem učení je potom $u(k)$. Výběr vektoru zpoždění je mnohem složitější než u přímého modelu a je dobré otestovat více možností. Při použití neuronové sítě, jako regulátoru, je vstupní vektor v podobě (7.15).

$$[y(k-2), y(k-1), w(k), u(k-2)] \tag{7.15}$$

Výstupem neuronové sítě je potom $u(k-1)$, což je hodnota akčního zásahu, kterou je nutné znát při regulaci.

Regulátor v podobě inverzního modelu byl implementován v simulinku. Schéma je podobné jako u přímého modelu jen jsou změněny vstupy a je dostupné v příloze VIII.. Počet použitých neuronů je stejný jako v minulém případě a to šest.

7.4 Porovnání několika typů regulátorů v simulaci

Jak již bylo psáno, kritéria pro porovnání jsou regulační kritéria, kritérium potřeby paměti a kritérium potřebných matematických operací.

7.4.1 Porovnání z hlediska regulace

Porovnání z hlediska regulace bylo provedeno již zmíněnými integrálními kritérii v tabulce. Grafy regulace jsou dostupné v příloze I a II. Porovnáno bude několik úseků, jako

změna akční veličiny odpovídající malé změně pólů a velká změna akční veličiny pro velkou změnu pólů. Dále budou uvedeny hodnoty adaptivního regulátoru při počáteční adaptaci. U každé tabulky jsou uvedeny změny pólů lineární soustavy při skoku. Pro testy byly nastaveny stejné póly pro všechny regulátory.

Tab. 7.1. Porovnání regulace pro skok w z 0 na 0,1, odpovídající Δp_z 0.06

Parametr	1DOF	NN (6N)	NN + PS (6N)	Adaptivní (1DOF)	Adaptivní (2DOF)
S_y	$4,25\times10^{-4}$	$5,44\times10^{-5}$	$1,01\times10^{-4}$	$2,76\times10^{-4}$	$4,43\times10^{-4}$
S_u	$8,94\times10^{-6}$	$5,08\times10^{-4}$	$9,87\times10^{-5}$	$2,55\times10^{-5}$	$9,87\times10^{-5}$
$\max y$	0,1	0,1022	0,1	0,1	0,1
$\min y$	0	0	0	0	0
$\max u$	0,333	0,712	0,383	0,333	0,333
$\min u$	0,128	0,335	0,226	0,118	0,025
překmit y	0%	2,2%	0%	0%	0%

V tabulce (Tab. 7.1) jsou vidět kritéria regulace skoku z 0 na 0,1. Při tomto skoku dojde jen k malé změně pólů lineární diskrétní soustavy a proto byl zvolen. Výsledkem je že si v této oblasti všechny regulátory regulují podobně. Lze zde spatřit jednu nevýhodu neuronové sítě a to, že došlo k překmitu a následně trvala regulační odchylka 0,0022. Důvodem je pravděpodobně malý počet vzorků v dané oblasti při učení. Současně si lze všimnout, že přidaný PS regulátor tento problém vykompenzoval. Dále je patrné, že neuronová síť dokončila regulaci nejrychleji podle S_y za cenu největšího akčního zásahu ze všech regulátorů. Oba adaptivní regulátory začínaly z odhadů blízkých prvnímu skoku.

Tab. 7.2. Porovnání regulace pro skok w z 0,1 na 0,4, odpovídající Δp_z 0.45

Parametr	1DOF	NN (6N)	NN + PS (6N)	Adaptivní (1DOF)	Adaptivní (2DOF)
S_y	0,0013	$4,63\times10^{-5}$	$8,96\times10^{-4}$	0,003	0,004
S_u	$7,41\times10^{-4}$	0,0148	0,0013	0,0052	0,0371
$\max y$	0,4006	0,4002	0,4	0,6807	0,5342
$\min y$	0,1	0,1	0,1	0,1	0,1
$\max u$	0,718	1,811	0,996	1,234	1,747
$\min u$	0,333	0,336	0,333	0,333	0
překmit y	0,2%	0,06%	0%	93,6%	44,7%

U skoku v tabulce (Tab. 7.2) dochází k největší adaptaci při tomto regulačním pochodu, zároveň probíhá největší změna pólů lineární diskrétní soustavy. Na tomto

skoku jsou zřetelné problémy adaptivních regulátorů v podobě velkého překmitu. Naopak běžný PS regulátor se strukturou 1DOF udělala jen velmi malý překmit, ale dlouho se ustaloval, proto má špatné S_y. Obě verze regulátoru umělé neuronové sítě jsou bez překmitu, ale umělá neuronové síť bez přídavného PS regulátoru, má ustálenou regulační odchylku 0,0002. Je také dobře vidět, že akční zásah všech regulátorů kromě (1DOF)PS a umělé neuronové sítě s PS byl o dost větší.

Tab. 7.3. Porovnání regulace pro skok w z 0,6 na 0,7, odpovídající Δp_z 0,02

Parametr	1DOF	NN (6N)	NN + PS (6N)	Adaptivní (1DOF)	Adaptivní (2DOF)
S_y	$1,19\times10^{-4}$	$5,28\times10^{-5}$	$1,01\times10^{-4}$	$5,82\times10^{-5}$	$2,37\times10^{-4}$
S_u	$8,66\times10^{-5}$	0,0016	$1,17\times10^{-4}$	$8,66\times10^{-4}$	$1,11\times10^{-4}$
$\max y$	0,7061	0,704	0,7	0,7013	0,7
$\min y$	0,6	0,601	0,6	0,6	0,6
$\max u$	0,9459	1,2881	1,0136	1,1839	0,9214
$\min u$	0,8176	0,8184	0,8184	0,8176	0,8176
překmit y	6,1%	4%	0%	1,3%	0%

Skon v tabulce (Tab. 7.3) byl zvolen z důvodu, že nedochází velké adaptaci a zároveň dochází k malé změně pólů lineární diskrétní soustavy. Bez překmitu byly jen umělá neuronová síť s PS regulátorem a Adaptivní 2DOF. Adaptivní 1DOF a umělá neuronová síť byly nejrychlejší regulátory za cenu velkých akčních zásahů.

7.4.2 Porovnání z hlediska regulace s poruchou na výstupu

Důležitým pohledem při porovnání je také odolnost proti poruchám. Proto v této kapitole budou zvolené regulátory prověřeny proti poruše v podobě bílého šumu na výstupu systému. Výkon bílého šumu byl zvolen na 0.00001W. Překmit nebude vyhodnocován z důvodu, že není úplně zřejmý při přidaném šumu u velkých překmitů bude upozorněno poznámkou. Výsledky regulace jsou uvedeny v příloze III a IV.

Na rozdíl od regulace bez poruchy, v tomto případě (Tab. 7.4) má adaptivní regulátor 1DOF velký relativní překmit a to 115%. Ostatní regulační pochody jsou velmi podobné regulaci bez poruchy.

Tab. 7.4. Porovnání regulace pro skok s poruchou w z 0 na 0,1, odpovídající Δp_z 0.06

Parametr	1DOF	NN (6N)	NN + PS (6N)	Adaptivní (1DOF)	Adaptivní (2DOF)
S_y	$2,82\times10^{-4}$	$1,61\times10^{-4}$	$2,4\times10^{-4}$	$7,63\times10^{-4}$	$5,51\times10^{-4}$
S_u	$2,79\times10^{-6}$	0,0064	0,0149	$8,51\times10^{-4}$	$1,12\times10^{-4}$
$\max y$	0,1325	0,1308	0,1336	0,2154	0,1306
$\min y$	0,0022	0,0036	0,0036	0	0
$\max u$	0,3708	0,6944	0,4956	0,6285	0,367
$\min u$	0,1398	0,2075	0,1467	0,095	0,0201

Tab. 7.5. Porovnání regulace pro skok s poruchou w z 0,1 na 0,4, odpovídající Δp_z 0.45

Parametr	1DOF	NN (6N)	NN + PS (6N)	Adaptivní (1DOF)	Adaptivní (2DOF)
S_y	0,0011	$6,29\times10^{-4}$	0,0011	0,0033	0,0027
S_u	0,0014	0,0271	0,0339	0,0153	0,0187
$\max y$	0,4532	0,4397	0,4476	0,5149	0,5374
$\min y$	0,0923	0,092	0,0865	0,0923	0,1048
$\max u$	0,8874	1,8052	0,99	0,937	1,0137
$\min u$	0,3362	0,3822	0,3413	0,2604	0,3351

V tabulce (Tab. 7.5) jsou patrné horší výsledky v porovnání s případem bez šumu, ale překmity u adaptace jsou nepatrně menší. Relativní překmity u obou metod byly okolo 50%. Dále se projevil zásadní překmit jen u PS regulátoru a to 45%.

Z výsledků v tabulce (Tab. 7.6) jde vidět, že všechny kritéria se zhoršily ale regulační pochod proběhl v pořádku. Jediný zásadní překmit nastal u PS regulátoru s 1DOF strukturou. Dále je také patrné, že kritérium S_u se zhoršilo zásadně u všech měření.

Je zřejmé, že všechny regulátory si dokázaly dobře poradit s chybou na výstupu. Oba regulátory s umělou neuronovou sítí se chovaly téměř stejně jako bez přidaného šumu.

7.4.3 Porovnání z hlediska potřeby paměti

Regulace přiřazením pólů regulační struktuře 1DOF bude využívat nejméně paměti ze všech uvedených metod. V tomto případě je třeba jen dvou minulých hodnot regulační odchylky a jedné minulé hodnoty akčního zásahu pro určení akčního zásahu v případě linearizace prvním řádem. Regulátor má celkem dva koeficienty.

V případě neuronové sítě je paměťová náročnost závislá na počtu vstupů, počtu výstupů a počtu skrytých neuronů. Pokud se označí počet vstupů N, počet neuronů

Tab. 7.6. Porovnání regulace pro skok s poruchou w z 0,6 na 0,7, odpovídající Δp_z 0,02

Parametr	1DOF	NN (6N)	NN + PS (6N)	Adaptivní (1DOF)	Adaptivní (2DOF)
S_y	$2,28\times10^{-4}$	$2,04\times10^{-4}$	$3,11\times10^{-4}$	$2,03\times10^{-4}$	$4,08\times10^{-4}$
S_u	0,0057	0,0078	0,0149	0,0057	0,0081
$\max y$	0,7406	0,7323	0,7	0,7310	0,7312
$\min y$	0,61	0,5996	0,5919	0,6	0,5925
$\max u$	0,9969	1,2922	1,1055	1,1958	1,0097
$\min u$	0,746	0,7249	0,6804	0,7269	0,7269

ve skryté vrstvě H a počet výstupů O, potom počet potřebných vah a hranic P pro použitou neuronovou síť lze vypočítat s pomocí rovnice (7.16).

$$P = N \cdot H + H \cdot O + H + O \tag{7.16}$$

Hodnota byla vypočítána pro síť s 6 neurony. Co se týče délky zpoždění, tak ta je stejná jako u 1DOF regulátoru.Pro adaptivní řízení s modelem prvního řádu lze odvodit rovnici (7.17), pro počet hodnot z minulého okamžiku výpočtu.

$$P = 2 \cdot N + N \cdot N \tag{7.17}$$

Kde N je délka regresoru a také počet parametrů, $N \cdot N$ potom vyjadřuje velikost kovarianční matice. Dále je potřeba započítat minulé hodnoty pro daný algoritmus řízení.

Tab. 7.7. Porovnání paměťových potřeb jednotlivých algoritmů

Typ regulace	Minulé hodnoty	Koef. regulace
1DOF Přiřazení pólů	3	2
Umělá neuronová síť (6N)	3	37
Umělá neuronová síť (6N) + PS	3 + 3	37
Adaptivní řízení (1DOF)	8 + 3	2
Adaptivní řízení (2DOF)	8 + 3	3

V tabulce (Tab. 7.7) u adaptivních metod je před plus uveden počet hodnot potřebných pro metodu nejmenších čtverců. U umělé neuronové sítě s PS regulátorem je za plus uveden počet minulých hodnot pro PS regulátor.

7.4.4 Porovnání z hlediska počtu matematických operací

Porovnání s hlediska matematických operací je pravděpodobně ještě zásadnější než z hlediska paměti. Metody jako adaptivní řízení a neuronové sítě využívají poměrně velké množství násobení a sčítání/odčítání oproti 1DOF přiřazení pólů. Porovnání bude

rozděleno na počet násobení/dělení a počet sčítání protože tyto operace na běžných mikropočítačích trvají různou dobu. Při implementaci na levný RISC procesor bude přičtení provedeno za jeden cyklus procesoru a násobní za několik cyklů, protože je implementováno sčítáním. Pokud je použit DPS nebo mikropočítač s ARM architekturou tak násobení lze provést v jednom cyklu procesoru.

Přiřazení pólů struktuře 1DOF je výpočetně jednoduché v každém kroku je potřeba vynásobit minulé hodnoty regulačními konstantami a následně vše sečíst pro výpočet hodnoty akčního zásahu.

V případě neuronové sítě to bude opět záviset na velikosti jednotlivých vrstev, při výpočtu je ignorovaný výpočet aktivační funkce. Výsledný počet operací se dá získat pomocí rovnice (7.18).

$$P = N \cdot H + H \cdot O \tag{7.18}$$

Počet součtů/odečítání v umělé neuronové síti je potom určen rovnicí (7.19)

$$P = N \cdot H + H \cdot O \tag{7.19}$$

Protože jde vždy o násobení a vektoru a matice výpočet je poměrně jednoduchý. Počet sčítání je vždy o jedno menší než násobení, ale je třeba přičíst ještě prahy.

Pro adaptivní řízení je výpočet počtu operací složitější, pro násobení/dělení lze odvodit vzorec (7.21)

$$P = 4N^2 + 3N + 3 \tag{7.20}$$

Kde N je délka regresoru. A pro sčítání/odčítání platí (7.21).

$$P = 4N^2 \tag{7.21}$$

K adaptivním regulátorům je dále třeba přičíst výpočet akčního zásahu a výpočet regulátoru v každém kroku. Je ale potřeba poznamenat že adaptaci lze vypnout v případě, že systém není příliš nelineární.

Tab. 7.8. Porovnání paměťových potřeb jednotlivých algoritmů

Typ regulace	Násobení/dělení	Sčítání/Odčítání
1DOF Přiřazení pólů	2	2
Umělá neuronová síť (6N)	30	30
Umělá neuronová síť (6N) + PS	30 + 2	30 + 2
Adaptivní řízení (1DOF)	31 + 2 + 2	16 + 2 + 2
Adaptivní řízení (2DOF)	31 + 3 + 5	16 + 3 + 4

U adaptivních metod v tabulce (Tab. 7.8) je navíc uveden počet operací pro výpočet akčního zásahu a počet operací pro výpočet regulátoru. U umělé neuronové sítě je uveden navíc počet výpočtů pro PS regulátor.

8 Ověření na reálném modelu

Po otestování v simulacích, přišel čas otestovat algoritmus na skutečném systému. Byl zvolen systém nádrže z důvodu toho, že je stabilní a tím se problém získávání dat značně zjednodušuje. Jak bude dále patrné, tato skutečná nádrž má mnohem více nelinearit.

8.1 Popis modelu nádrží

Model nádrží je tvořen třemi nádržemi, několika ventily a dvěma čerpadly. Celý systém je uzavřený a voda je čerpána z sběrné nádrže. Všechny nádrže jsou v sekvenci propojeny ventily. První s druhou a druhá s třetí. Každá nádrž má také jeden ventil jako odtok do sběrné nádrže s vodou. Čerpadla mají přítok do první a poslední nádrže.

Ovládání modelu a sběr dat je prováděn přes Real-Time toolbox. Ventily a čerpadla je možné nastavovat pomocí ovládacího panelu, nebo v Simulinku. Pro čerpadlo existuje volba manuální režim a automatický režim. V manuálním režimu je čerpáno podle pozice potenciometru na ovládacím panelu a při automatickém režimu je hladina v nádrži u čerpadla držena na maximální hladině. Data o výšce hladiny nádrží jsou získána pomocí tlakového senzoru, jehož signál je dále potřeba zpracovat pro vhodné využití. Ventily je možné otevírat, zavírat nebo s nimi nemanipulovat. Ovládány jsou signálem -1,1 v simulaci, který potom převede karta počítače na signál pro řízení ventilu, nebo přepínačem na ovládacím panelu. Ventily mají signál o svojí pozici, který je před využitím dále zpracovat. Mají taky možnost ověření koncových poloh.

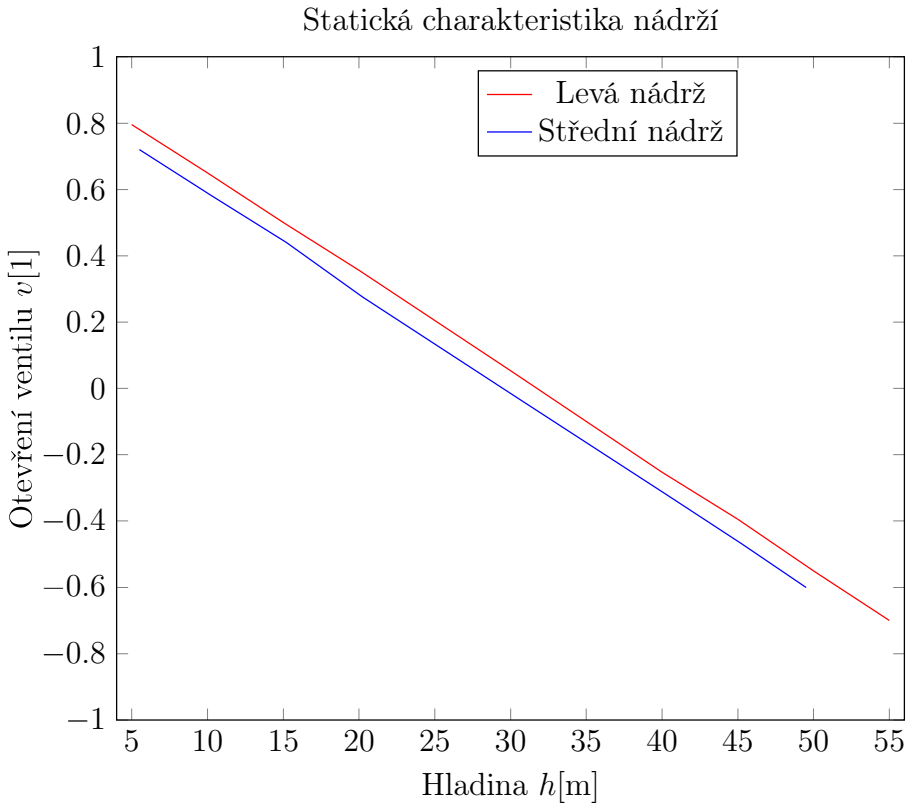
Obr. 8.1. Model nádrží v laboratoři

8.2 Návrh experimentu

Experiment bude prováděn na systému jedné nádrže a to střední nádrže modelu. Čerpadlo do první nádrže bude v automatickém režimu a první nádrž bude po celou dobu trvání experimentu přibližně na maximu. Regulace bude prováděna ventilem mezi první a druhou nádrží. Ventil mezi druhou nádrží a sběrnou nádrží bude částečně uzavřený aby zpomalil odtok z prostřední nádrže. Sběr dat bude prováděn na ventilu použitém k regulaci a také bude sledována výška hladiny v prostření nádrži.

8.3 Zpracování signálů do vhodné podoby

Prvním krokem pro ovládání modelu bylo zpracování signálů na modelu do vhodné podoby. Nejdříve byla určena závislost výstupu senzorů na ustálené hladině v nádržích.



Obr. 8.2. Statická charakteristika nádrží

Z grafů (Obr. 8.2) je patrné, že obě nádrže jsou si velmi podobné a vztah mezi otevřením ventilu a hladinou je lineární. Rovnice levé nádrže je $v_L = -0,03h_L + 0,9498$, rovnice prostřední nádrže je $v_S = -0,03h_S + 0,8884$. Rovnice pro výpočet hladin v simulačním schématu jsou (8.1) a (8.2).

$$h_L = -\frac{v_L - 0,9498}{0,03} \tag{8.1}$$

$$h_S = -\frac{v_S - 0,8884}{0,03} \tag{8.2}$$

Dalším krokem bylo určení toho jak je možné řídit ventily. Byly zjištěny hodnoty signálu u krajních pozicích ventilu a následně převedeny na hodnotu od 0 do 1 pro jednoduší práci s tímto signálem. Dalším důvodem je taky to, že ventily byly ve skutečnosti uzavřeny nebo plně otevřeny předtím, než dosáhli krajních hodnot a krajní hodnoty byly menší než 1. Hodnoty v tabulce (Tab. 8.1) jsou výsledky pokusu pro určení signálu ventilů.

Tab. 8.1. Naměřené hodnoty ventilů

Ventil	min	cca. 50%	max
č.1	0,522	-0,06	-0,697
č.4	0,493	-0,075	-0,715
č.5	0,489	-0,1	-0,67

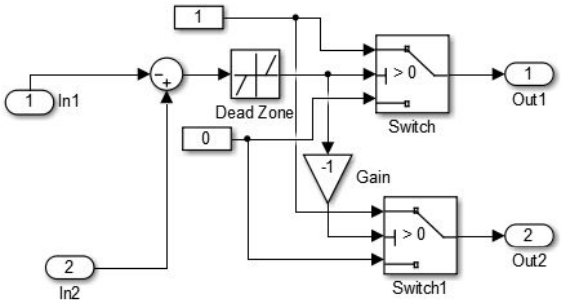
Z této tabulky lze získat rovnice, které převádí mezi naměřenými hodnotami do intervalu 0 až 1. Rovnice pro tuto konverzi jsou v podobě (8.3), eq:vent2 a eq:vent3.

$$v_{1N} = -\frac{v_1 - 0,522}{1,219} \tag{8.3}$$

$$v_{4N} = -\frac{v_4 - 0,493}{1,208} \tag{8.4}$$

$$v_{5N} = -\frac{v_5 - 0,489}{1,159} \tag{8.5}$$

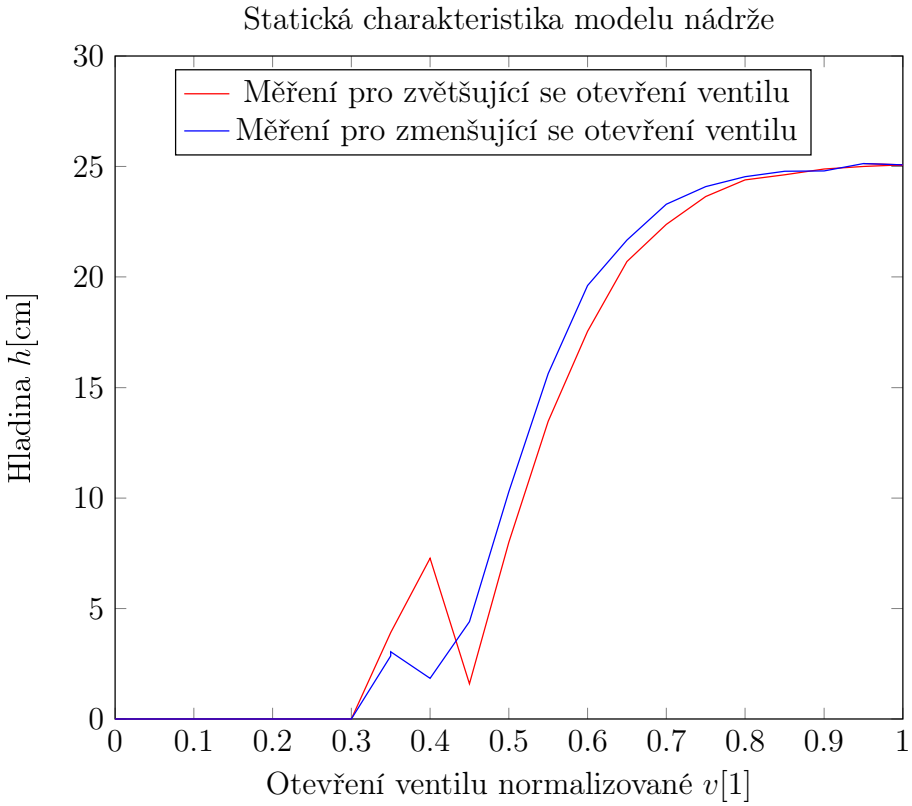
Ventil č. 1 slouží jako propojení mezi levou a střední nádrží, slouží tedy k regulaci při experimentu. Ventil č. 4 slouží k odtoku ze střední nádrže a při experimentu je nastaven na konstantní hodnotu. Ventil č. 5 je odtok z levé nádrže a v rámci experimentu je uzavřen. Dále bylo potřeba vybudovat schéma pro řízení polohy ventilu, ventily totiž umožňují pouze otevírání a uzavírání, to je na obrázku (Obr. 8.3).



Obr. 8.3. Simulink ovládání ventilů

8.4 Statická charakteristika

Prvním krokem měření byl průzkum statických vlastností systémů. To bylo provedeno naměřením statické charakteristiky (Obr. 8.4). Charakteristika byla změřena jak pro otevírání, tak pro uzavírání ventilu.

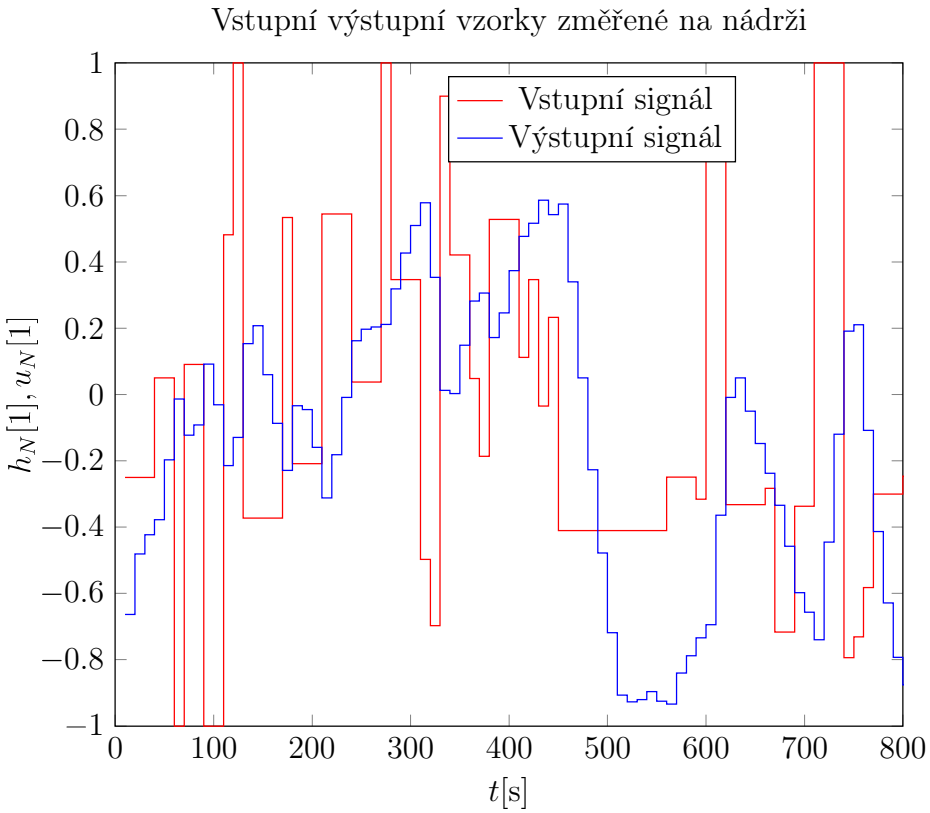


Obr. 8.4. Statická charakteristika získána měřením na reálném modelu

Z naměřené statické charakteristiky na obrázku (Obr. 8.4) je patrné, že systém je silně nelineární. V oblasti otevření ventilu 0 až 30% je mrtvé pásmo. Také je patrná nelinearita saturace okolo 25cm. Dále je možné si všimnout, že systém má hysterezi, kde zmenšující se akční zásah vykazuje menší hodnotu ustálené hladiny, než rostoucí akční zásah. V neposlední řadě je patrná zvláštní nelinearita v oblasti 30% až 45% otevření ventilu. Tato nelinearita bude velmi problematická z důvodu, že dané hladiny lze dosáhnout více způsoby akčních zásahů.

8.5 Sběr dat a učení sítě

Sběr dat byl proveden způsobem velmi podobným jako v simulaci. Bylo vygenerováno několik signálů typu „skyline“ z různými vzorkovacími frekvencemi a hodnotami α. Následně byly vybrány ty, které dobře identifikovali dynamické vlastnosti soustavy. Na následujícím obrázku je ukázka získaných již normalizovaných hodnot do rozsahu ± 1 pro vzorkovací periodu $10s$.



Obr. 8.5. Vstupní a výstupní vzorky pro učení umělé neuronové sítě

Po nasbírání dat byl naučen inverzní model k systému nádrže metodou obecného učení. Bylo využito skryté vrstvy s 10 neurony.

8.6 Zhodnocení regulace

Pro zhodnocení regulace byla použita opět integrální regulační kritéria, minima, maxima a relativní překmit. Porovnání na reálném modelu nádrže bylo provedeno na regulátou PS založeném na přiřazení pólů struktuře 1DOF, adaptivním regulátoru se strukturou 1DOF, umělou neuronovou sítí a umělou neuronovou sítí s dodatečným regulátorem PS. Regulátor 2DOF nebyl k porovnání použit, protože nedokázal regulovat porovnatelně rychle s ostatními regulátory. Všechny regulátory měly opět nastavené stejné póly. Výsledky regulace jsou uvedeny v příloze V a VI. Jednotlivé skoky byly opět zvoleny s úmyslem ukázat určité informace. První skok byl vybrán pro porovnání jak si regulátory poradí s nezvyklou linearitou. Druhý skok byl vybrán z důvodu, že při něm probíhala adaptace adaptivního regulátoru a poslední skok představuje ukázku běžné regulace.

Tab. 8.2. Porovnání regulace pro skok w z 0 na 5

Parametr	1DOF	NN (10N)	NN + PS (10N)	Adaptivní (1DOF)
S_y	$1,4722$	$0,6078$	$1,1188$	$0,1962$
S_u	$0,0012$	$0,0319$	$0,0225$	$6,22\times10^{-4}$
$\max y$	4,8636	5,0413	5,2222	0,51887
$\min y$	0	0	0	0
$\max u$	0,7227	4,9334	4,9376	0,4106
$\min u$	0	0	0	0
rel. přek. y	0%	0%	0%	0%

Podle výsledků tabulky (Tab. 8.2) a pohledu na regulační pochod v příloze V. a VI. je zřejmé, že nelinearita mezi 30% až 45 % způsobila zásadní problémy regulátoru PS a adaptivní regulátor měl taky problémy. Mimo toto pásmo oba regulátory fungovaly dobře, ovšem za cenu velkých akčních zásahů. To je ovšem také důvod správné regulace v dané oblasti.

Tab. 8.3. Porovnání regulace pro skok w z 5 na 10

Parametr	1DOF	NN (10N)	NN + PS (10N)	Adaptivní (1DOF)
S_y	$0,7661$	$0,8382$	$0,9967$	$0,5477$
S_u	$0,0011$	$0,2123$	$0,188$	$0,0014$
$\max y$	10,1373	10,1747	10,167	10,8949
$\min y$	4,7101	4,5586	4,9387	4,7101
$\max u$	0,9288	5,6446	5,0607	0,9999
$\min u$	0,3321	0,1253	0	0,3917
rel. přek. y	2,746%	3,494%	3,34%	17,8%

Při tomto skoku (Tab. 8.3) se nejvíce projevila adaptace 1DOF regulátoru, který měl velký relativní překmit. Ostatní regulátory prakticky bez překmitu. U regulace jsou opět patrné velmi vysoké hodnoty akčního zásahu u obou regulátorů, které využívají umělé neuronové sítě. Naopak v tomto ohledu jde vidět velmi dobrý výsledek u PS a adaptivního 1DOF regulátoru.

Tab. 8.4. Porovnání regulace pro skok w z 18 na 12

Parametr	1DOF	NN (10N)	NN + PS (10N)	Adaptivní (1DOF)
S_y	$0,964$	$0,8868$	$1,101$	$0,6859$
S_u	$5,96\times10^{-4}$	$0,2105$	$0,1978$	$0,0019$
$\max y$	18,2363	18,2622	18,1355	18,0829
$\min y$	11,4788	11,6836	11,8168	11,4788
$\max u$	0,4926	0,4664	0,1832	0,6927
$\min u$	0	0	0	0
rel. přek. y	3,8%	4,3%	2,16%	1,33%

Tento skok byl vybrán z důvodu, že by odpovídal běžné regulaci ve vyšší hladině. Z výsledků regulace v tabulce (Tab. 8.4) je patrné, že všechny regulátory v tomto případě měly velký překmit akční veličiny, který byl omezený na nule. Ostatní kritéria jsou prakticky totožné.

Z výsledků na reálném modelu je patrné, že regulátory s umělými neuronovými sítěmi, které jsou učené obecným učením, můžou generovat poměrně velké překmity. Z hlediska regulované veličiny byly po překonání první nelinearity výsledky všech regulátorů porovnatelné.

ZÁVĚR

Teoretická část byla věnována základům v oblasti umělých neuronových sítí jako neuron, aktivační funkce a učení neuronu. Následně teoretická část pokračuje informacemi o vrstvě umělé neuronové sítě a umělé neuronové síti jako celku. V této kapitole byl taky uveden algoritmus backpropagation. Byly také uvedeny možnosti rekurentních umělých neuronových sítí.

V následujících kapitolách teoretické části byly probrány možnosti aplikace umělých neuronových sítí v oblastech návrhu regulátoru. Nejdříve byly popsány základy identifikace a její postup. Dále byly uvedeny možnosti využití neuronových sítí v oblasti identifikace, kde byly probrány vhodné signály pro identifikaci s využitím umělých neuronových sítí. Následně byla popsána základní kritéria pro ověření kvality modelu neuronových sítí jako korelační funkce, histogram chyb a regrese. Krátce byla probrána příprava dat pro umělé neuronové sítě. Po objasnění problémů v oblasti identifikace byly uvedeny vybrané používané algoritmy pro řízení s využitím umělých neuronových sítí. U každého algoritmu byl uveden postup aplikace na reálný systém a také nějaké výhody, nevýhody daného algoritmu.

Teoretická část začala seznámením s průběhem implementace a stanovením benchmark systému pro pokusy s regulací. Popis implementace byl doplněn ukázkami kódu z MATLAB. Následně byla na benchmark systému ověřena funkčnost regulace a bylo provedeno porovnání s jinými typy regulátorů. Porovnání bylo z několika hledisek, jako počet operací při výpočtu, velikost nutné paměti, integrálních regulačních kritérií a překmitu regulované veličiny. Následně jsem se věnoval aplikaci vytvořených programů na skutečný systém.

Výsledkem této práce je hlavně porovnání různých typů regulátorů s umělou neuronovou sítí. Pro porovnání byly vybrané jiné typické regulátory, které by mohly být použity pro neznámý model se slabší nelinearitou. Při porovnání z hlediska potřeby paměti pro uchování informace jednoznačně skončila neuronová síť jako poslední a ostatní typy regulátorů mají podobně malou potřebu paměti. Porovnáním z hlediska operací potřebných výpočtu bylo zjištěno, že nejvíce operací využije adaptivní řízení následované umělou neuronovou sítí. Je ale důležité poznamenat, že adaptace může být v ustálených stavech vypnuta a počet operací se následně redukuje jen na výpočet akčního zásahu. Při pohledu z hlediska regulace lze dojít k výsledku, že neuronová síť s přidaným PS regulátorem dosáhla nejlepších výsledků z hlediska regulované veličiny. Z hlediska akčního zásahu nejlepšího výsledku dosáhla regulace adaptivním regulátorem s přiřazením pólů 2DOF struktuře, pokud není počítán skok při kterém došlo k adaptaci. Velmi dobrý výsledek dosáhl i standardní PS regulátor a to jak v oblasti akční veličiny tak v oblasti regulované veličiny. Nevýhodou tohoto regulátoru naopak byl nekonzis-

tentní regulační pochod v různých oblastech řízení. Samotná neuronová síť dosáhla až příliš rychlé regulace, což způsobilo velký akční zásah. Dalším problémem neuronové sítě je že nedokázala úplně dosáhnout žádané veličiny a vždy byla viditelná alespoň velmi malá regulační odchylka. To je způsobeno tím, že i když umělá neuronová síť využívá minulých hodnot není v pravém smyslu zpětnovazební regulátor, proto po přidání PS regulátoru který zpomalí regulaci neuronové sítě a odstraní ustálenou regulační odchylku byly výsledky mnohem lepší. Po tomto ověření byly současně regulátory ověřeny za přítomnosti poruchy na výstupu systému. Výsledky byly velmi podobné, až na PS regulátor ten dosahoval poměrně velkých překmitů při všech změnách žádané veličiny. Naopak obě adaptivní řízení dosáhli lepších výsledků než v případě bez poruchy.

Při aplikaci na reálný systém bylo dosaženo velmi podobných výsledků jako v simulaci. Navíc se ukázalo, že neuronová síť si umí dobře poradit s nelinearitami typu mrtvá zóna a omezení. PS regulátor dosáhl velmi dobrých výsledků až na oblast skoku žádané veličiny z 0 na 5 cm. Jeho regulace nebyla úplně konzistentní, někdy vykazoval překmit. Adaptivní regulátor získaný přiřazením pólů struktuře 1DOF měl dobré výsledky, ale měl také několik překmitů. Umělá neuronová síť s PS dosáhla dobrých výsledků ale při její regulaci došlo k jednomu překmitu. To by šlo vylepšit volbou pomalejšího regulátoru PS. Samotná neuronová síť v tomto případě podala asi nejlepší výsledek z pohledu sledování žádané veličiny.

Z výsledků v této práci lze usoudit, že algoritmy umělých neuronových sítí jsou v oblasti řízení dobře použitelné, ale za cenu velké spotřeby paměti a složitosti výpočtu. Regulace s umělými neuronovými sítěmi je dostatečná jen v jednoduchých lineárních nebo nelineárních systémech. Ve složitějších nelineárních případech je vhodné k umělé neuronové síti připojit více regulátorů se zpětnou vazbou. Velkou výhodou je to, že umožňuje aplikaci běžných lineárních regulátorů, což v této práci vedlo k velmi dobrým výsledkům. Problémem je ale, že mimo pracovní oblast, kde nebyly sbírány data nebude umělá neuronová síť regulovat správně, což by mohlo vést ke zničení regulované soustavy. Dále bylo také zjištěno že neuronová síť učená obecným učením má poměrně velký překmit akčního zásahu, což omezuje oblast použití tohoto trénování.

Programy z této práce je možné použít k návrhu jednoduchých regulátorů využívajících umělé neuronové sítě. Schémata vytvořená v simulink lze použít pro simulaci těchto sítí, popřípadě regulaci s využitím Real Time Toolbox. Tato práce a vytvořené programy mohou být použity pro implementace jiných zmíněných regulátorů a k implementaci specializovaného učení.

Výsledky regulace jsou v podobě grafů uvedeny v příloze práce. Veškeré kódy, data a simulační schémata jsou na přiloženém CD.

SEZNAM POUŽITÉ LITERATURY

[1] HAGAN, Martin T., Howard B. DEMUTH, Mark H. BEALE a Orlando DE JESÚS. *Neural network design*. Vyd. 2. Boston: PWS Pub., 2014. ISBN 9780971732117.

[2] NORGAARD, Magnus, Ole RAVN, Niels K. POULSEN a L. K. HANSEN. *Neural networks for modelling and control of dynamic systems: a practitioner's handbook*. New York: Springer, 2000. ISBN 1852332271.

[3] RASCHKA, Sebastian. *Python Machine Learning*. Birmingham: Packt Publishing, 2015. ISBN 9781783555130.

[4] BOBÁL, Vladimír. *Adaptivní a prediktivní řízení*. Zlín: Univerzita Tomáše Bati, 2007. ISBN 978-80-7318-662-2.

[5] ŠÍMA J., NERUDA R.: *Teoretické otázky neuronových sítí*. Vyd. 1. Praha, 1996: Matfyzpress, 390 s., ISBN 80-85863-18-9.

[6] KŘIVAN, Miloš. *Úvod do umělých neuronových sítí*. Vyd. 3., přeprac. Praha: Oeconomica, 2014, 44 s. ISBN 978-80-245-2024-7.

[7] MATLAB: Neural Network Toolbox. *MathWorks* [online]. Massachusetts: MathWorks, 1994- [cit. 2017-03-29]. Dostupné z: `https://www.mathworks.com/products/neural-network.html`.

[8] MATLAB. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-03-29]. Dostupné z: `https://en.wikipedia.org/wiki/MATLAB`.

[9] MATLAB: Design NARMA-L2 Neural Controller in Simulink. MathWorks [online]. Massachusetts: MathWorks, 1994- [cit. 2017-04-06]. Dostupné z: `https://www.mathworks.com/help/nnet/ug/design-narma-l2-neural-controller-in-simulink.html`.

[10] BRÁZDIL, Michal. Výpočet obecného řídícího zákona prediktivního řízení pro systému druhého řádu bez omezení. Posterus [online]. Bratislava, 2011 [cit. 2017-04-20]. Dostupné z: `http://www.posterus.sk/?p=11147`

[11] Intelligent control. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-04-25]. Dostupné z: `https://en.wikipedia.org/wiki/Intelligent_control`

[12] Butterworth filter. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-04-25]. Dostupné z: `https://en.wikipedia.org/wiki/Butterworth_filter`

[13] BOBÁL, Vladimír. Identifikace systémů. Zlín: Univerzita Tomáše Bati ve Zlíně, 2009. ISBN 978-80-7318-888-7.

[14] Lipschitz continuity. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-04-25]. Dostupné z: `https://en.wikipedia.org/wiki/Lipschitz_continuity`

[15] HRUBÝ, Martin. Využití neuronové sítě při řízení hydraulického modelu [online]. [cit. 2017-04-25]. Dostupné z: `http://akce.fs.vsb.cz/1997/Asr97/sbornik/hruby/hruby.htm`

[16] NOSKIEVIČ, Petr. Modelování a identifikace systémů. Ostrava: Montanex, 1999. ISBN 8072250302.

[17] ŠPAČEK, Ondřej. Simulační modely vybraných řízených systémů [online]. Zlín, 2015 [cit. 2017-05-18]. Dostupné z: http://hdl.handle.net/10563/34249. Bakalářská práce. UTB.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

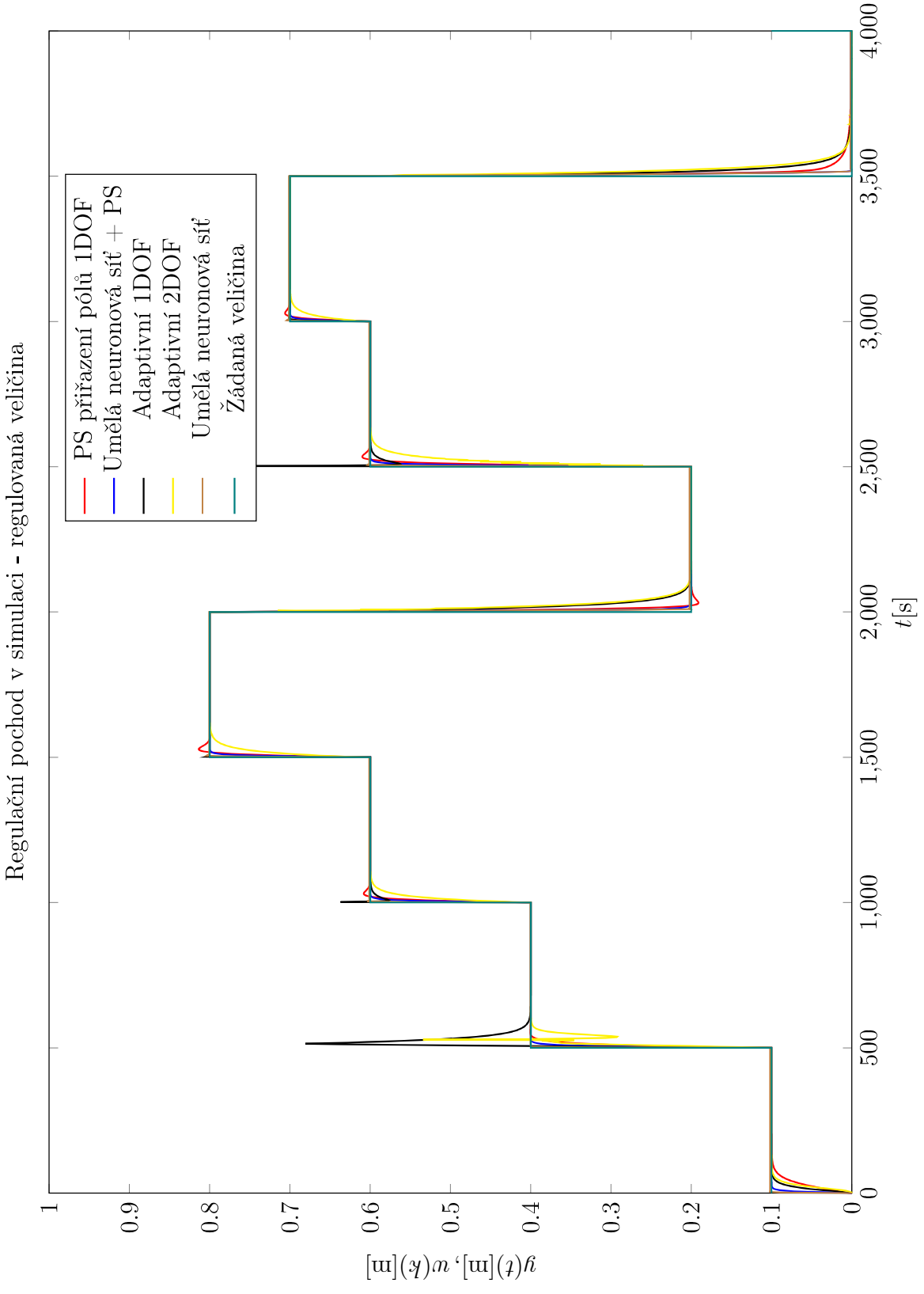
$\boldsymbol{a}$	Výstup neuronové sítě
$\boldsymbol{b}^i$	Práh i-té vrstvy umělé neuronové sítě
b	Práh
e_s	Bílý šum
F	kritérium
f^i	Aktivační funkce vrstvy
$G(z^{-1})$	Diskrétní přenos soustavy
MIMO	Multi input multi output
n	Výsledek produktu vah se vstupními vzorky a součet s prahem
SISO	Single input single output
R	Korelační funkce
S_u	Integrální kritérium akčního zásahu
S_y	Integrální kritérium regulované veličiny
$\boldsymbol{t}$	Vektor cílů učení
$u(k)$	Akční veličina/akční zásah
$\boldsymbol{X}$	Vstupní vzorek
x_i	i-tý prvek vstupního vzorku
$y(k)$	Regulovaná veličina
w_i	i-tá váha neuronu
$w(k)$	Žádaná veličina
$\boldsymbol{W}^i$	Váhy i-té vrstvy umělé neuronové sítě
Δu	Přírůstek akčního zásahu
Θ	Regresní vektor
Φ	Vektor parametrů

SEZNAM OBRÁZKŮ

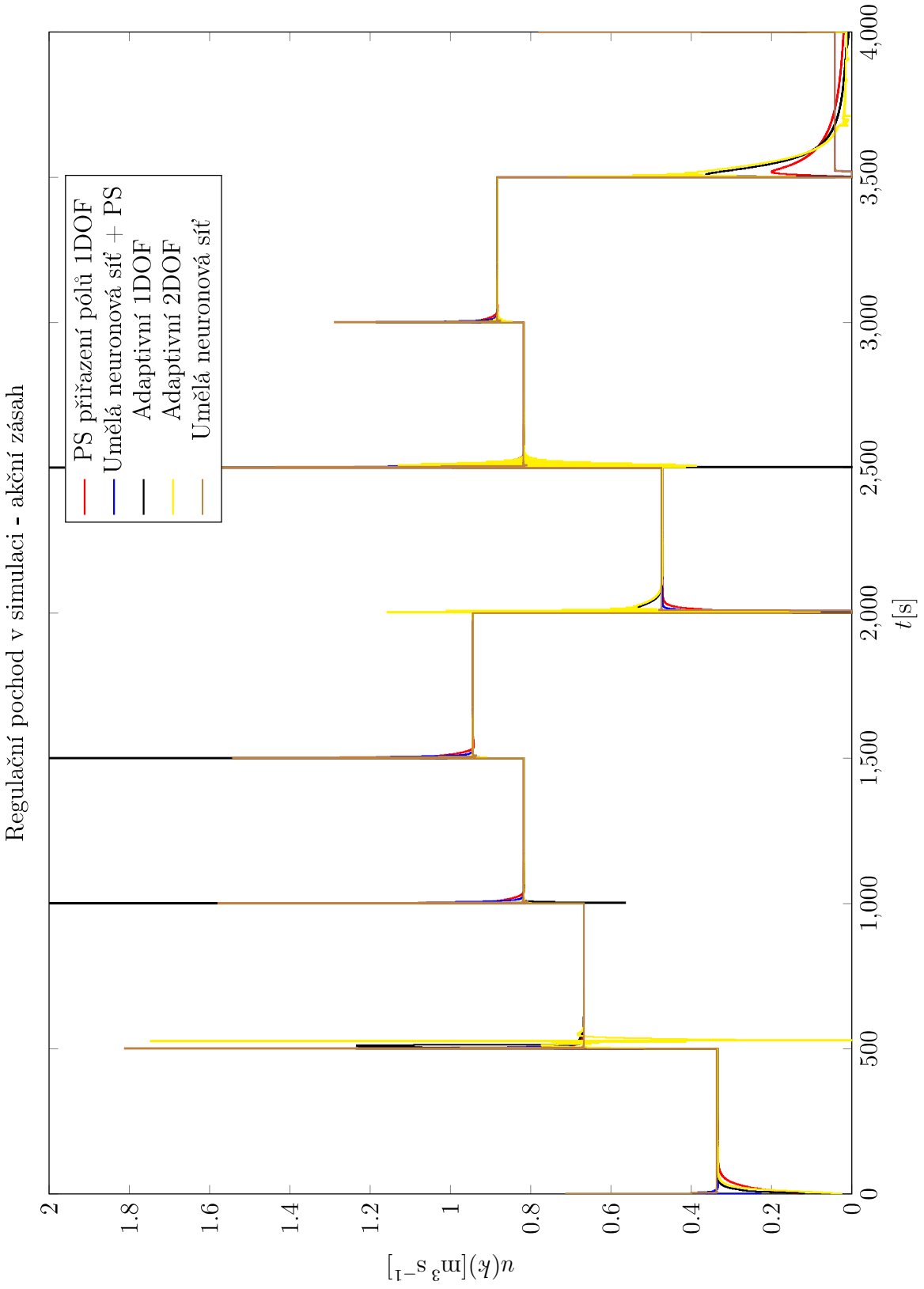
SEZNAM TABULEK

SEZNAM PŘÍLOH

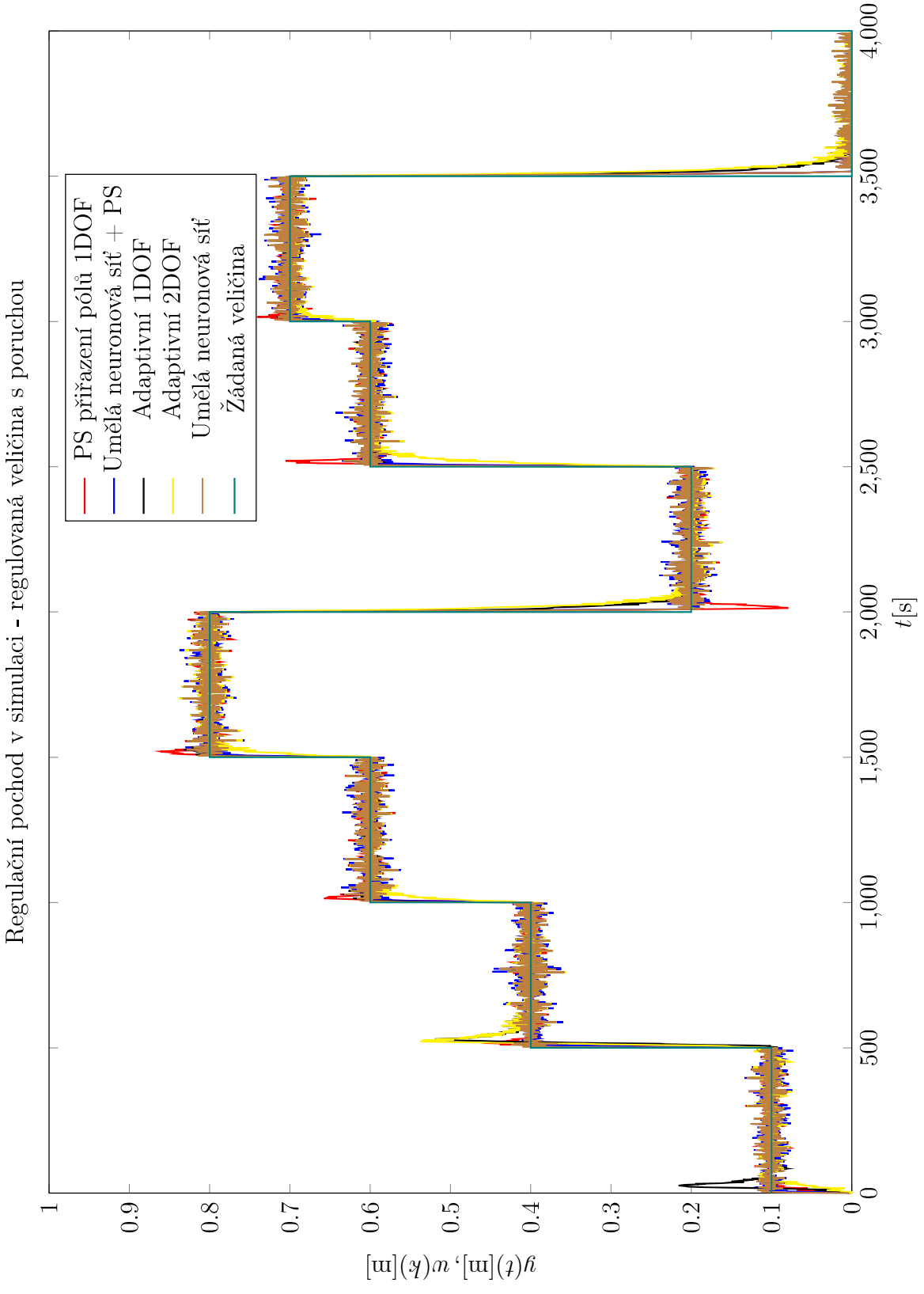
PŘÍLOHA P I. BENCHMARK SYSTÉM SIMULACE - REGULOVANÁ VELIČINA



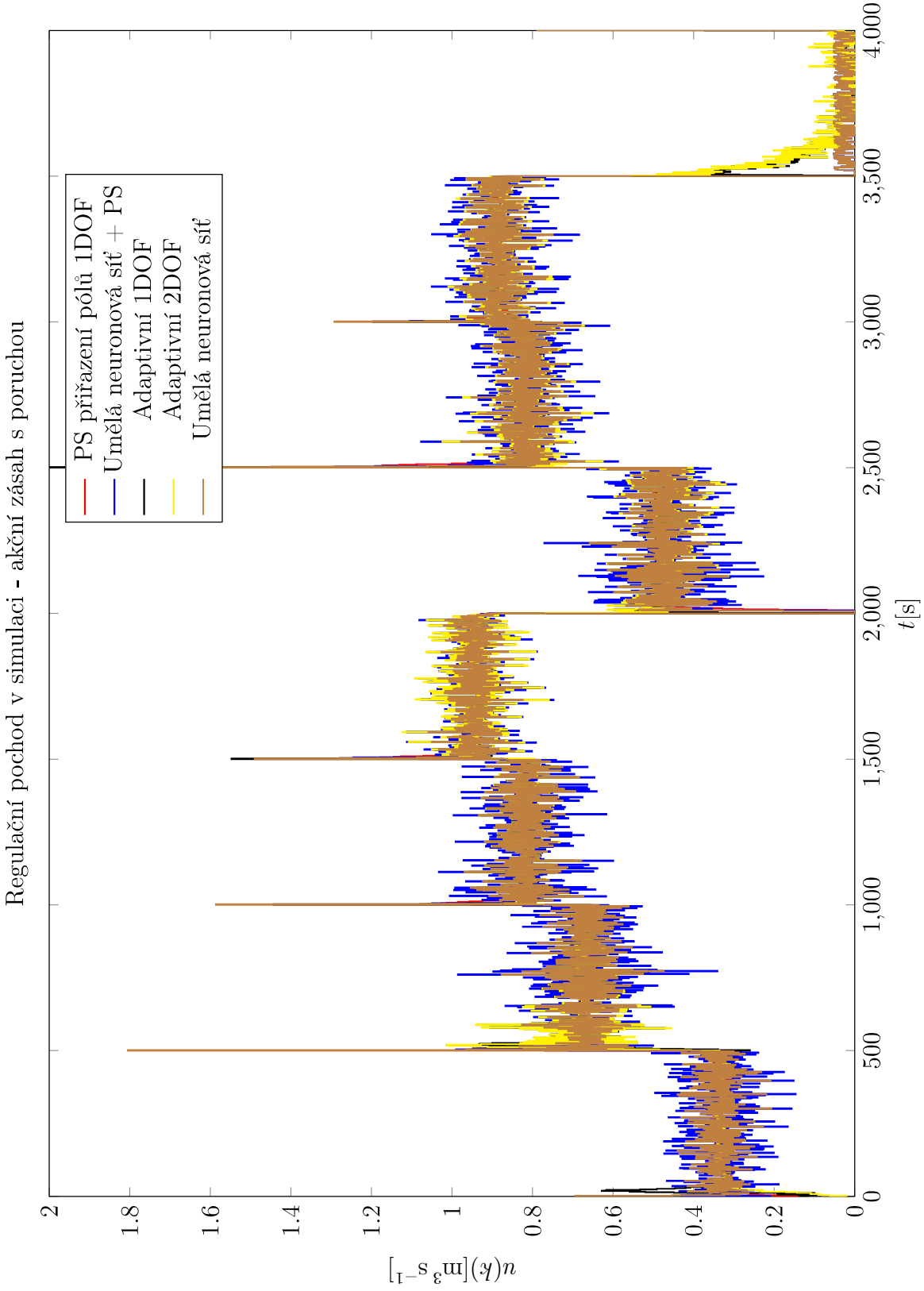
PŘÍLOHA P II. BENCHMARK SYSTÉM SIMULACE - AKČNÍ ZÁSAH



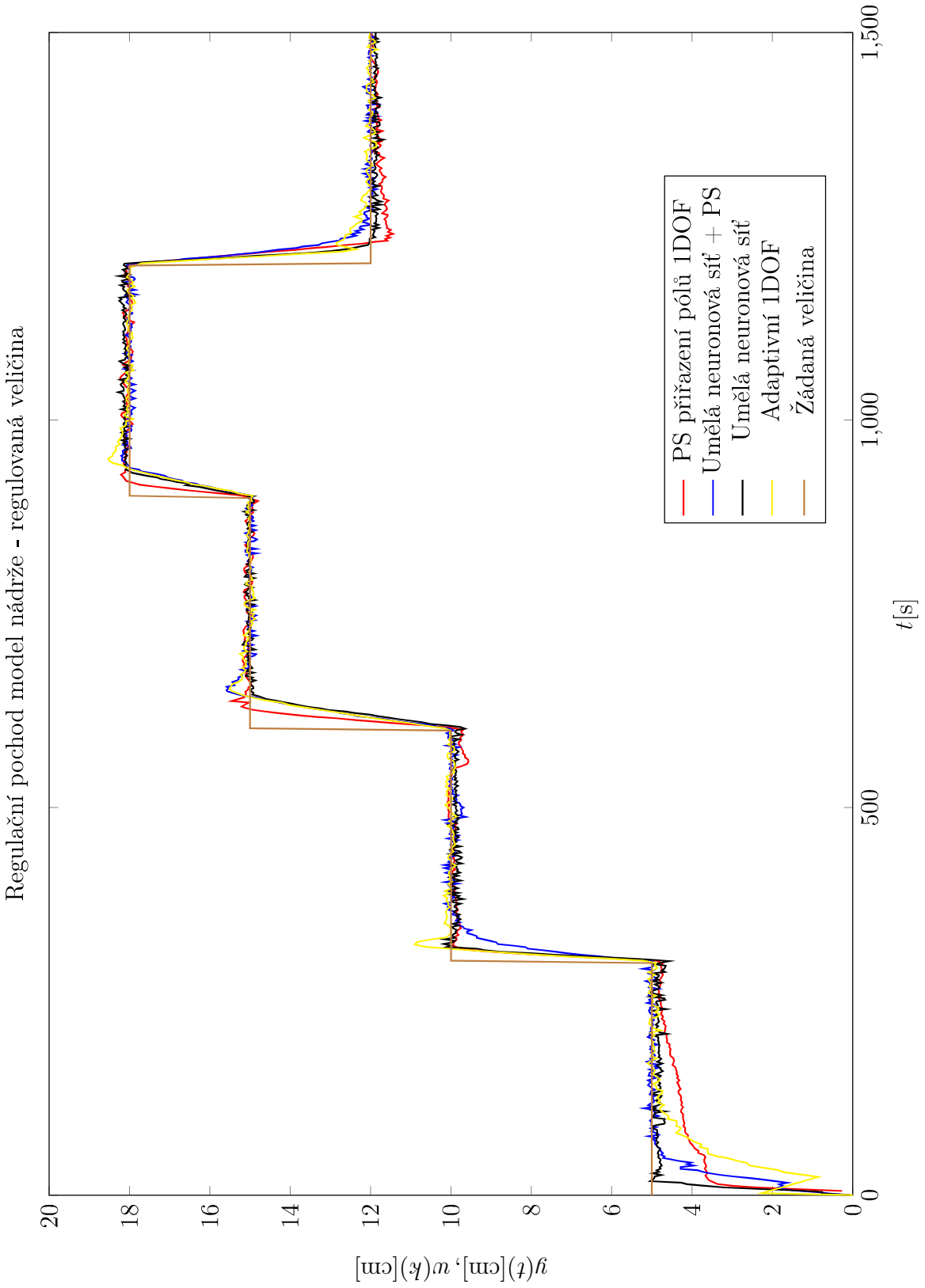
PŘÍLOHA P III. BENCHMARK SYSTÉM SIMULACE - REGULOVANÁ VELIČINA S PORUCHOU



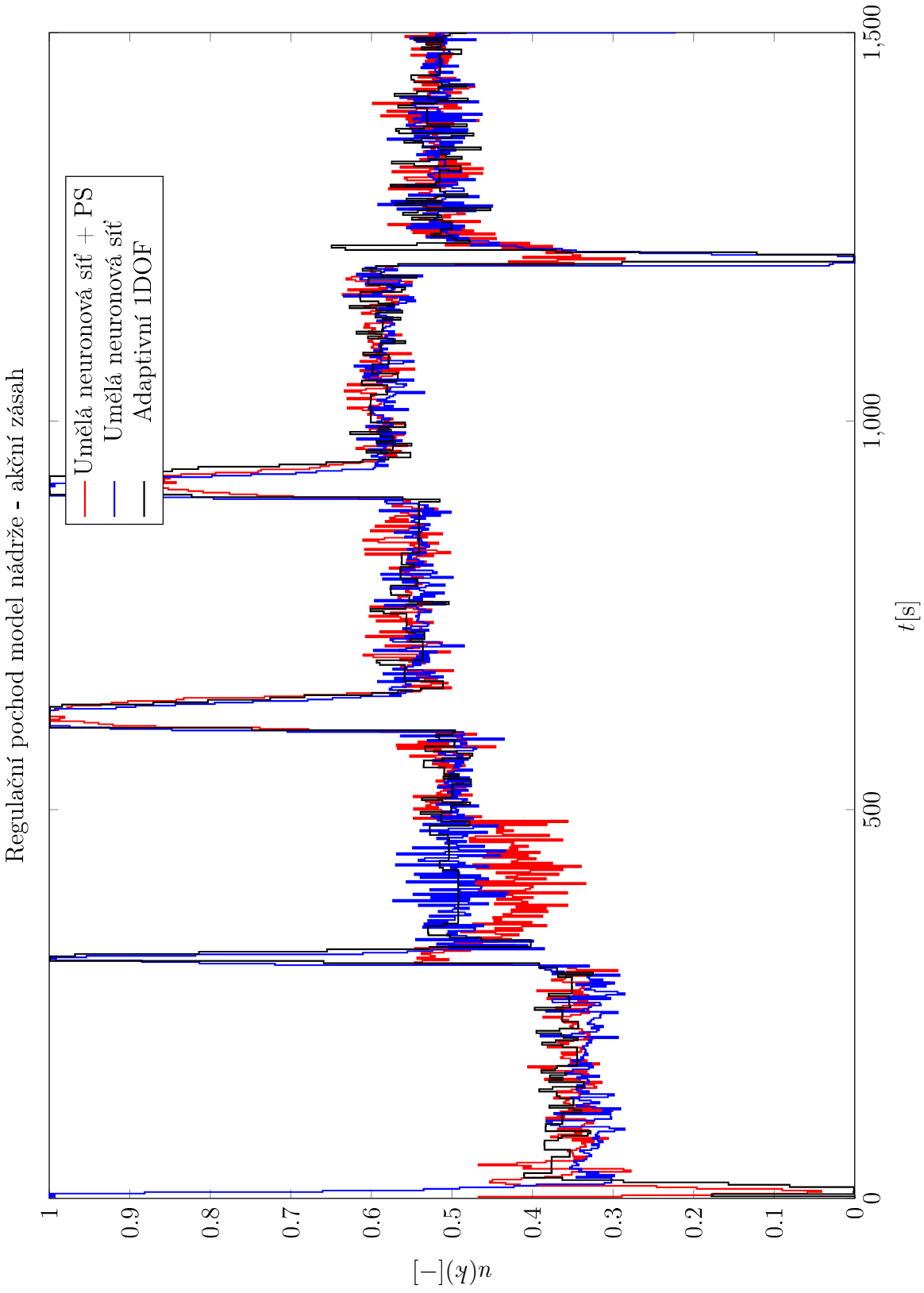
PŘÍLOHA P IV. BENCHMARK SYSTÉM SIMULACE - AKČNÍ ZÁSAH S PORUCHOU



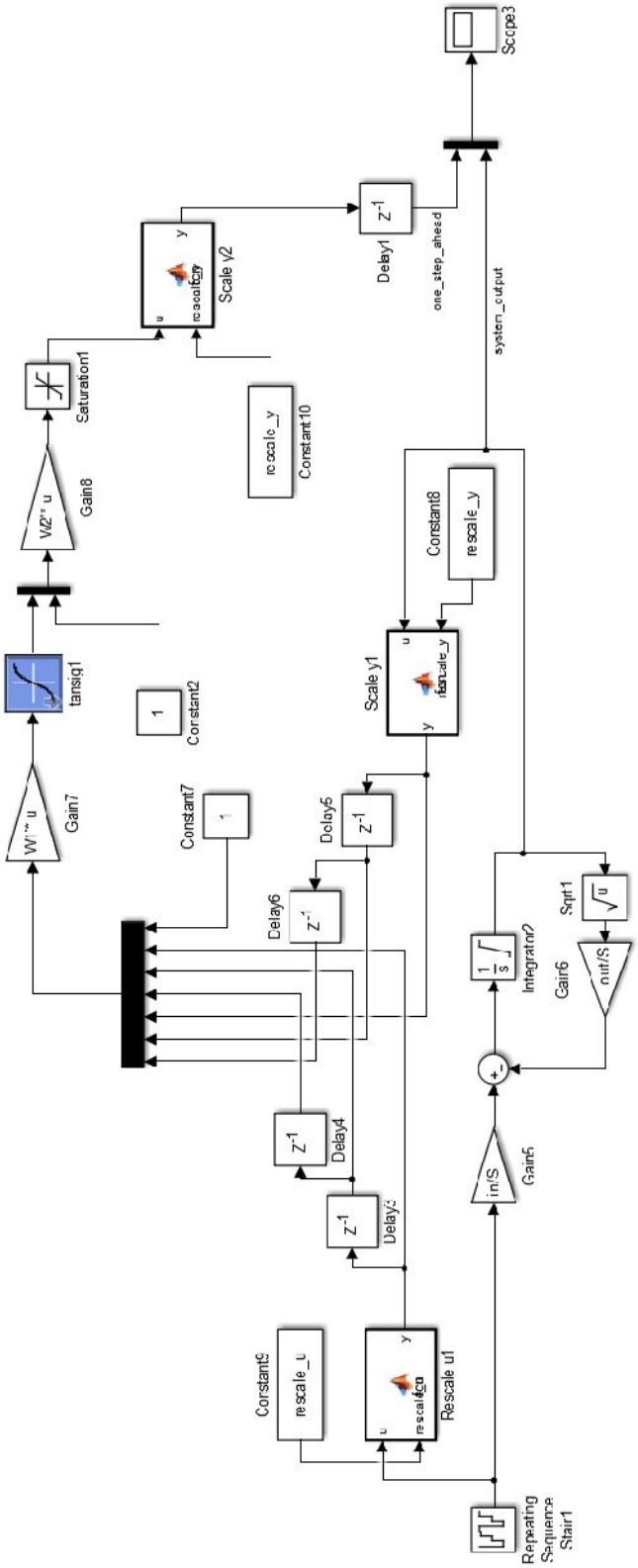
PŘÍLOHA P V. MODEL NÁDRŽÍ - REGULOVANÁ VELIČINA



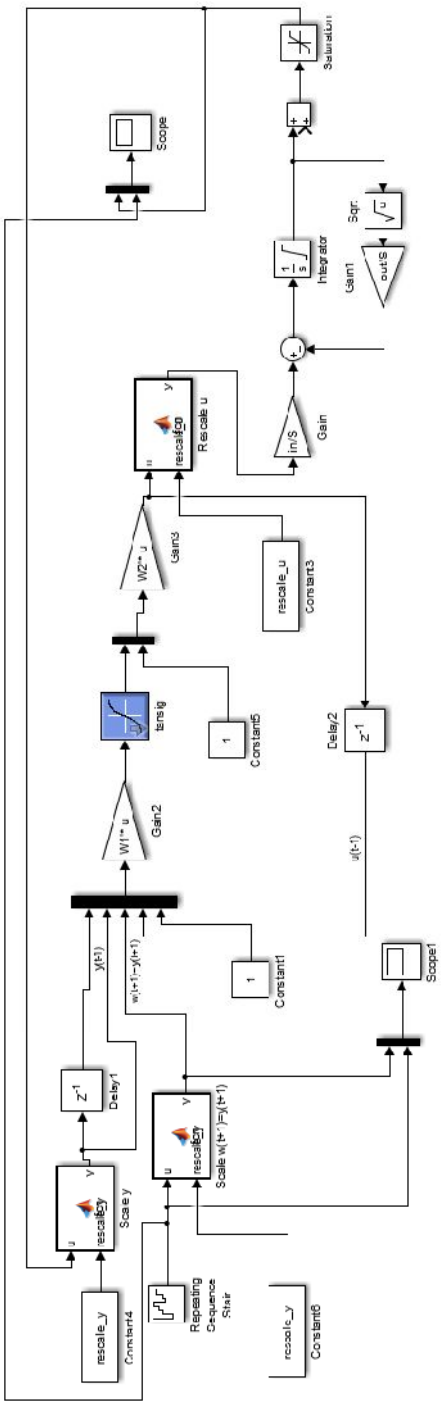
PŘÍLOHA P VI. MODEL NÁDRŽÍ - AKČNÍ ZÁSAH



PŘÍLOHA P VII. SIMULAČNÍ SCHÉMA DOPŘEDNÉ SÍTĚ



PŘÍLOHA P VIII. SIMULAČNÍ SCHÉMA INVERZNÍHO MODELU



PŘÍLOHA P IX. OBSAH CD

```
MATLAB_SKRIPTY_DATA
|--BENCHMARK_SYSTEM
|          |--SIMULACE_REGULACE
|    |--STAT_PRECH_LIN_CHAR
|          |--UCENI_SITE
|--NADRZ_LAB
     |--NAMERENA_DATA_REGULACE
     |--NAMERENA_DATA_STATICKA
     |--POUZITA_SCHEMATA_MATLAB2006
```

- `BENCHMARK_SYSTEM` – obsahuje veškeré skripty a schémata MATLAB/Simulink použité u benchmark systému
  - `SIMULACE_REGULACE` – obsahuje ukázku simulace regulace, před ukázkou je třeba spustit skript `sim_nadrz_inicializace.m`, následně je možné spustit simulaci `sim_nadrz_porucha.slx`
  - `STAT_PRECH_LIN_CHAR` – obsahuje skripty použité pro vykreslení grafů statické char., přechodové char. a ukázku linearizace v několika bodech a vývoj pólů
  - `UCENI_SITE` – obsahuje všechny skripty a simulace potřebné ke generování dat a naučení sítě, učení lze spustit `trenovani_neuronove_site_inicializace.m`
- `NADRZ_LAB` – obsahuje veškeré skripty, data a schémata MATLAB/Simulik použité na reálném systému nádrže
  - `NAMERENA_DATA_REGULACE` – obsahuje naměřená data na reálném systému a jejich vykreslení
  - `NAMERENA_DATA_STATICKA` – obsahuje naměřená data statické charakteristiky a jejich vykreslení
  - `POUZITA_SCHEMATA_MATLAB2006` – složka obsahuje použitá schémata v laboratoři