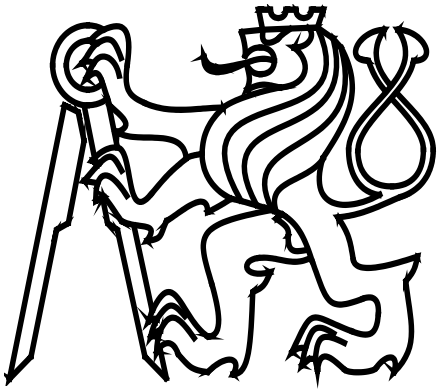
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta strojní



DIPLOMOVÁ PRÁCE

Nelineární Neuro-regulátor pro úlohy automatického řízení

Obor: Přístrojová a řídící technika

2008 Ladislav SMETANA

Anotační list

Jméno autora:	Ladislav SMETANA
Název DP:	Nelineární Neuro-regulátor pro úlohy automatického řízení
Anglický název:	Nonlinear Neuro-controller for automatic control laboratory system
Rok:	2008
Obor studia:	Přístrojová a řídící technika
Ústav/odbor:	Přístrojová a řídící technika
Vedoucí DP:	Ing. Ivo Bukovský, Ph.D.

Bibliografické údaje:

počet stran	63
počet obrázků	44
počet tabulek	1
počet příloh	1 (elektronické - CD)

Klíčová slova: neuron, neuronová jednotka, nelineární systémy, HONNU, QNU, adaptabilní řízení, Batch-Training, Levenberg-Marquardt, Metoda falešných sousedů

Keywords: neuron, neural unit, nonlinear systems, HONNU, QNU, adaptive control, Batch-Training, Levenberg-Marquardt, False Nearest Neighbors

Anotace: Cílem práce je návrh, studie, a implementace nelineárního Neuro-regulátoru pro řízení laboratorního nelineárního dynamického systému batyskafu. Diskrétní dynamická kvadratická neuronová jednotka je použita pro identifikaci systému. Diskrétní statická kvadratická neuronová jednotka jako stavový zpětnovazební adaptivní regulátor zároveň s konvenčním proporciálním regulátorem. K adaptaci Neuro-regulátoru slouží gradientová metoda učení Back-Propagation. Výsledná regulace Neuro-regulátorem přinesla podstatné zlepšení při regulaci v celém rozsahu regulované veličiny. Tato práce otevírá novou cestu a možnosti adaptivního řízení.

Abstract: The goal of the thesis is to design, study, and implement a nonlinear adaptive neuro-controler for control of the laboratory nonlinear dynamic system bathyscaph. The discrete dynamic quadratic neural unit is used for system identification. The static quadratic neural unit a state feedback adaptive controller in parallel with the conventional proportional controller as is used for system control. To adapt the controller, the gradient dynamic back propagation learning rule is implemented. In results, the implemented neuro-controler crucially improves the control process in the whole range of process variables. This thesis introduces a novel promising way of nonlinear adaptive control.

Prohlášení

Prohlašuji, že jsem diplomovou práci na téma „Nelineární Neuro-regulátor pro úlohy automatického řízení“ vypracoval samostatně pod vedením Ing. Ivo Bukovského, Ph.D. a použil pouze literární prameny uvedené v bibliografii.

V Praze dne 10. 12. 2008 Ladislav Smetana

Poděkování

Touto cestou bych velice rád poděkoval vedoucímu mé diplomové práce Ing. Ivu Bukovskému, Ph.D. za vstřícný přístup, inspirativní vedení, cenné rady a připomínky, kterými mi byl po celou dobu tvorby této diplomové práce nápomocen.

Obsah

1 Zadání……9
2 Cíl diplomové práce……9
3 Úvod……..10
3.1 Popis soustavy batyskafu 11
3.1.1 URO batyskafu s PID regulátorem 13
3.2 Nelineární neuronové jednotky s nelinearitami vyšších řádů 16
3.3 Statické HONNU 18
3.3.1 Trénování statických HONNU – Batch-Training 19
3.3.2 Levenberg-Marquardt algoritmus (LM) 19
3.3.3 Adaptace statických HONNU 21
3.4 Diskrétní Dynamické HONNU 23
3.5 HONNU jako zpětnovazební stavový adaptabilní regulátor 23
4 Řešení……24
4.1 Volba nelineárního modelu – QNU 24
4.1.1 False Nearest Neighbors - FNN 27
4.2 Identifikace batyskafu pomocí QNU 32
4.2.1 Batch-Training (statická QNU) 32
4.2.2 Adaptace - dynamická metoda 35
4.2.3 Porovnání statické (LM) a dynamické (BP) tréninkové metody 39
4.3 QNU jako nelineární stavový regulátor pro BATYSKAF 40
4.3.1 Simulace regulace soustavy Neuro-regulátorem (teoretická) 43
4.3.2 Reálná regulace soustavy Neuro-regulátorem (praktická) 46
5 Výsledky……...53
6 Závěr……56
Literatura……58
Přílohy……60

Použité termíny a symboly

BP	…	Back-Propagation (algoritmus učení)
CNU	…	Cubic Neural Unit
LNU	…	Linear Neural Unit
DNU	…	Dynamic Neural Unit
HONNU	…	Higher-Order Nonlinear Neural Unit (kde řád závisí na úrovni řádu polynomu nelinearity agregační funkce)
QNU	…	Quadratic Neural Unit
LM	…	Levenberg – Marquardt (algoritmus učení)
BT	…	Batch-Training
FNN	…	False Nearest Neighbors (medota falešných blízkých sousedů)
DDSOE-QNU	…	Discrete Dynamic-Second-Order-Extended QNU

1 Zadání

Nelineární Neuro-regulátor pro úlohy automatického řízení:

Navrhněte adaptabilní nelineární neuronovou jednotku, která bude aproximovat nelineární model reálné soustavy v co nejširším rozsahu regulované veličiny.

Navrhněte a aplikujte vhodnou nelineární neuronovou jednotku (kvadratickou, kubickou) s případnou vhodnou modifikací nelinearit jako nelineární zpětnovazební stavový Neuro-regulátor soustavy.

Navrhněte vhodný postup pro počáteční seřízení nelineární neuronové jednotky jako stavového Neuro-regulátoru. Aplikujte na vhodné soustavě v laboratoři AŘ s využitím prostředí Matlab Simulink.

2 Cíl diplomové práce

Cílem této práce je výběr a otestování vhodné varianty neuronové jednotky jako regulátoru nelineární soustavy batyskafu. Volba způsobu učení neuronové jednotky. Porovnání Neuro-regulátoru s klasickým regulátorem (například PID).

3 Úvod

Již od počátků vývoje prvních zařízení nebo složitějších strojů, a s tím spojenými procesy jejich výroby, měli lidé potřebu tyto postupy automatizovat a tím dále zlepšovat jejich kvalitu. Přirozená snaha člověka o to, aby si co nejvíce ulehčil život a usnadnil práci vedla, a stále více vede ruku v ruce se zdokonalováním veškerých procesů, k vývoji a aplikaci různých sofistikovaných řídících systémů. Samotný obor automatizace zaznamenal největší rozmach s nástupem výpočetní techniky, ale jeho kořeny sahají mnohem dál do historie. Jako příklad jednoho z prvních automatických regulátorů můžeme zmínit například mechanický odstředivý regulátor tlaku, používaný hojně ve spojení s parními stroji.

Moderní automatizace zná nespočet různých regulátorů, ať už regulátory mechanické, klasické regulátory popsané notoricky známou proporciální, integrační nebo derivační složkou, různé odvozené regulátory, regulátory a postupy regulace speciálně určené pro regulaci většího počtu veličin nebo regulátory nelineární. Žádný regulátor není dokonalý a při aplikaci na reálnou soustavu, která obsahuje různé reálné poruchy nebo dokonce nelinearity, nedosáhneme uspokojivé regulace a tím ani žádaného řídícího procesu. Neustálý postup ve vývoji nových myšlenek, teorií a jejich aplikací v praxi umožňuje řešit úlohy, které ještě před několika lety byly mimo možnosti řídících systémů.

Tato práce se bude věnovat možnosti využití Neuronové jednotky vyššího řádu HONNU (Higher Order Nonlinear Neural Unit) jako regulátoru reálné nelineární soustavy. Pro regulaci soustavy v celém rozsahu regulované veličiny je použití klasického lineárního regulátoru (například regulátoru PID) nedostatečné. Správně zvolená a vycvičená neuronová jednotka má veškeré předpoklady k lepšímu a přesnějšímu zvládnutí regulace takovéto nelineární soustavy v celém jejím rozsahu regulované veličiny.

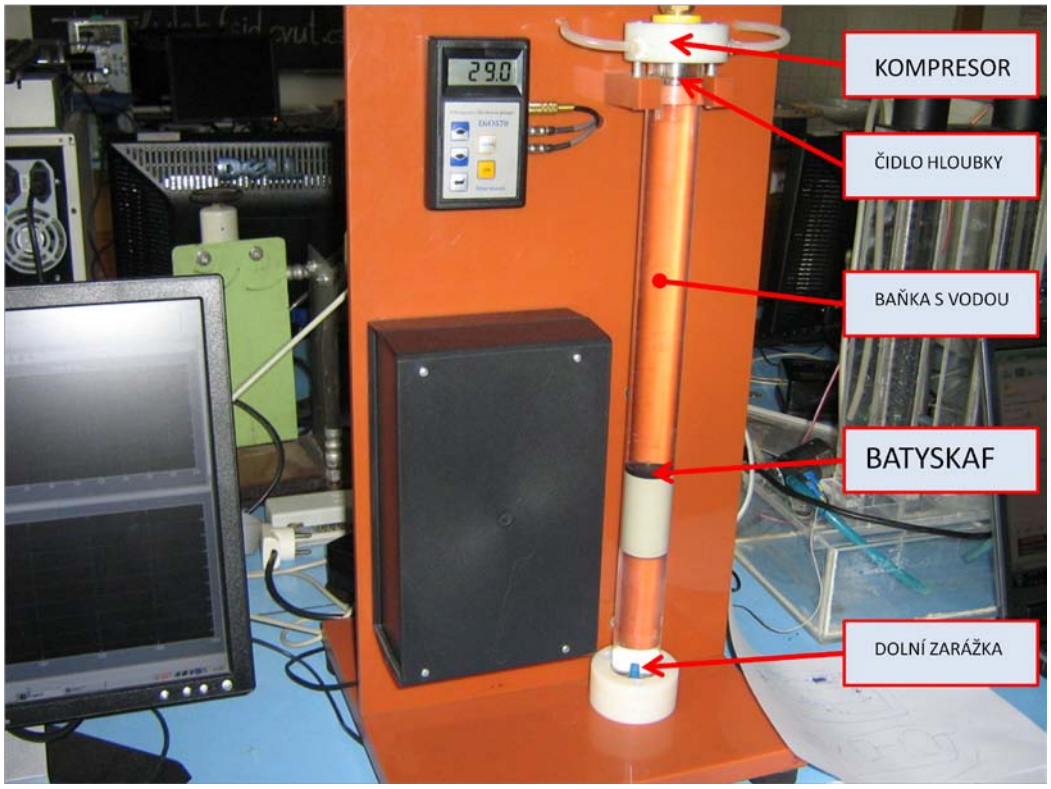
Batyskaf je nelineární soustava druhého řádu [13]. Nelinearita je dána jednak měnícím se tlakem v závislosti na hloubce batyskafu ve vodním sloupci, ale také nelinearitami čidel, kompresoru, případně dalšími prvky soustavy.

Klasické regulátory mají s takovou regulací problémy, protože nezvládají dobře nelinearitu soustavy. Projeví se to například tak, že seřízený regulátor bude dobře regulovat v malé hloubce a špatně v hloubce větší nebo naopak. Klasický regulátor není schopen pokrýt celý regulovaný rozsah s uspokojivými výsledky regulace.

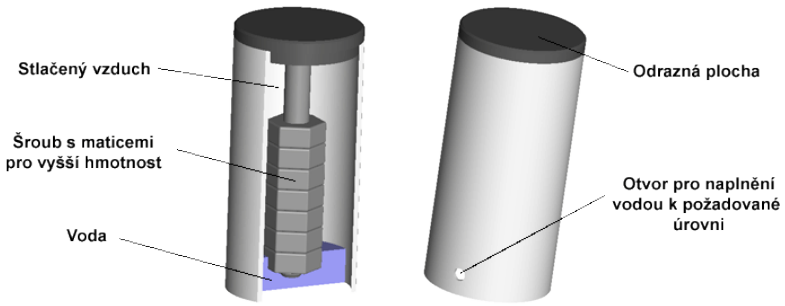
Volbou správného typu neuronové jednotky pro identifikaci dynamiky soustavy batyskafu a vhodné neuronové jednotky pro použití jako stavového zpětnovazebního Neuro-regulátoru bychom měli docílit, že takovýto regulátor dokáže zastínit celý rozsah regulované veličiny.

3.1 Popis soustavy batyskafu

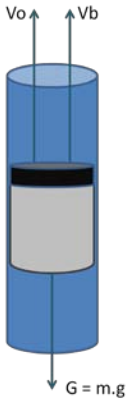
Laboratorní úloha Batyskaf se skládá ze skleněného válce naplněného vodou, dutého válečku (batyskafu - kelímek obrácený dnem vzhůru), kompresoru a čidel. Váleček (batyskaf) je umístěn ve válci s vodním sloupcem. Dutina batyskafu obsahuje závaží, zajišťující snazší pohyb směrem dolů, a vzduchovou bublinu zajišťující pohyb vzhůru (Obr. 2). Kompresor zajišťuje požadovaný tlak nad hladinou vodního sloupce. Na samotný batyskaf působí gravitační síla G a vztlakové síly V_o (vztlaková síla od vzduchové bubliny) a V_b (vztlaková síla tělesa batyskafu) - Obr. 3. Celá soustava pak funguje tak, že se zvyšujícím se tlakem nad hladinou dochází ke stlačování vzduchové bubliny uvnitř batyskafu a v důsledku toho batyskaf klesá. Naopak dojde-li ke snížení tlaku nad hladinou, vzduchová bublina se zvětší a dojde k tomu, že batyskaf začne stoupat. Ve vyvážené poloze se celková vztlaková síla rovná síle gravitační.



Obr. 1 – Popis laboratorní úlohy Batyskaf



Obr. 2 – Vnitřek batyskafu (převzato z [16])



Obr. 3 – Síly působící na batyskaf

Akční veličina v soustavě batyskafu reprezentuje otočení serva ventilu, které svým otočením reguluje přívod tlaku nad hladinu hodnoty natočení serva jsou udávány v radiánech. Fyzikálně je ovšem onou akční veličinou zapříčiňující pohyb batyskafu měnící se tlak udávaný v kPa.

Matematicko-fyzikální model popisuje nelineární rovnice 2. řádu, rovnice (1)[13]. Rovnice (1) může být dále rozšířena o jednotlivé matematické popisy motorů kompresoru, součinitel odporu proti pohybu válečku v kapalině, různá dopravní zpoždění a další zpřesnění.

$$\ddot{y}(t) = g - \dot{y}(t)\frac{k}{m} - \frac{V_b \cdot \rho \cdot g}{m} - \frac{V_o \cdot \rho \cdot g}{m} \cdot \frac{p_a}{p_a + p_x(t) + g \cdot y(t)} \qquad (1)$$

, kde g je gravitační zrychlení

m je hmotnost válečku

k je koeficient tlumení kapaliny

ρ je hustota kapaliny

V_b je objem válečku

V_o je objem vzduchové bublinky (v horní poloze, kde působí jen atmosférický tlak)

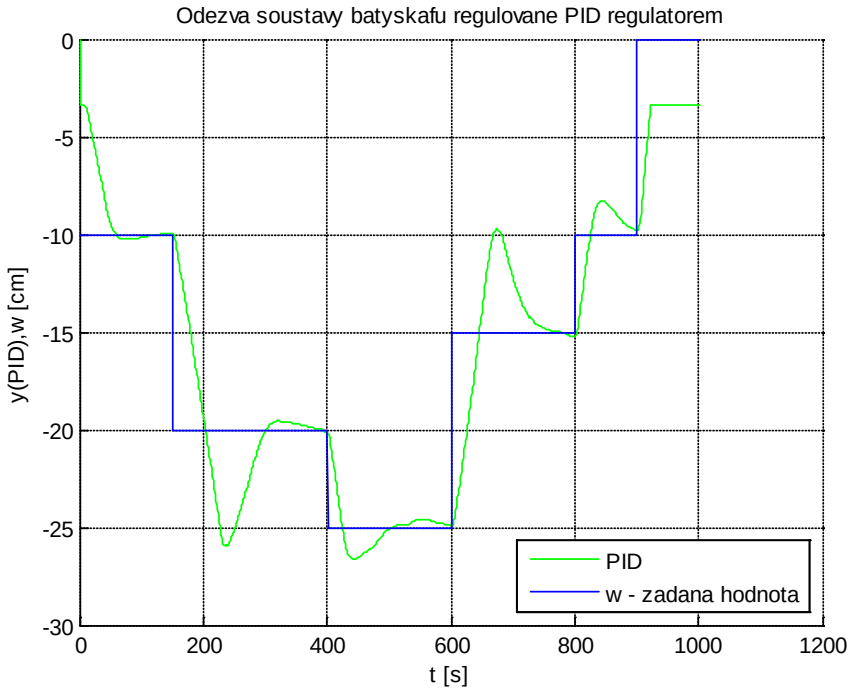
P_a je atmosférický tlak

P_x je tlak dodaný kompresorem

3.1.1 URO batyskafu s PID regulátorem

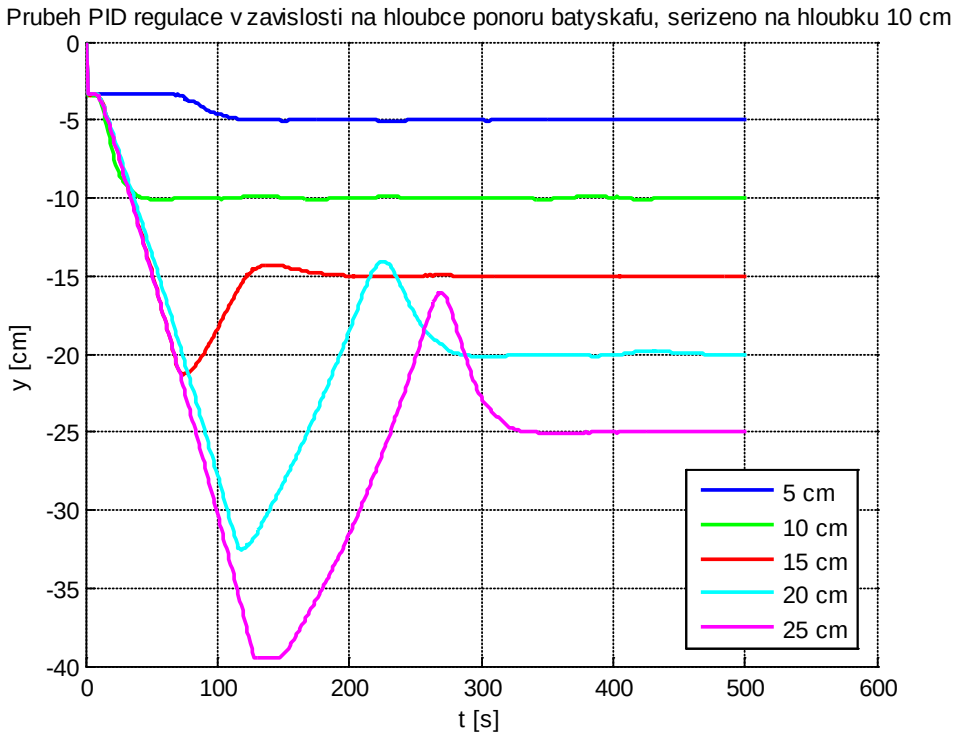
Pro regulaci hloubky batyskafu s použitím klasického PID regulátoru lze dosáhnout poměrně přesné regulace jen v úzkém okolí pracovní hloubky, pro kterou PID regulátor vyladíme. U soustavy batyskafu nezáleží jen na regulační hloubce, ale i na výchozí poloze batyskafu, ze které požadujeme změnu jeho polohy (počáteční stav), ale také na délce změny jeho dráhy. Experimenty prokázaly, že se batyskaf s PID regulátorem chová jinak, má-li se ponořit například na hodnotu hloubky 10 cm z maximální horní polohy (3,3 cm) nebo má-li se na tuto hodnotu dostat z hloubky 5 cm. Z toho vyplývá, že pro tuto nelineární soustavu s PID regulátorem hraje značnou roli i velikost změny žádané veličiny. Na Obr. 4 je zobrazena odezva soustavy batyskafu na nastavenou požadovanou hodnotu hloubky a její skokové změny. Je dobře vidět, že se soustava s vyladěným PID regulátorem pro změnu hloubky o 10 cm má při regulaci na tuto hloubku jen minimální překmit. Ponoření batyskafu do větší hloubky s takto vyladěným PID regulátorem ukázalo, že PID regulátor zvládá regulaci velice špatně.

Obdobně je tomu při pohybu batyskafu směrem vzhůru. Špatná schopnost regulace klasického PID regulátoru je dána nelinearitou soustavy a rozdílnou rychlostí stoupání a klesání batyskafu.

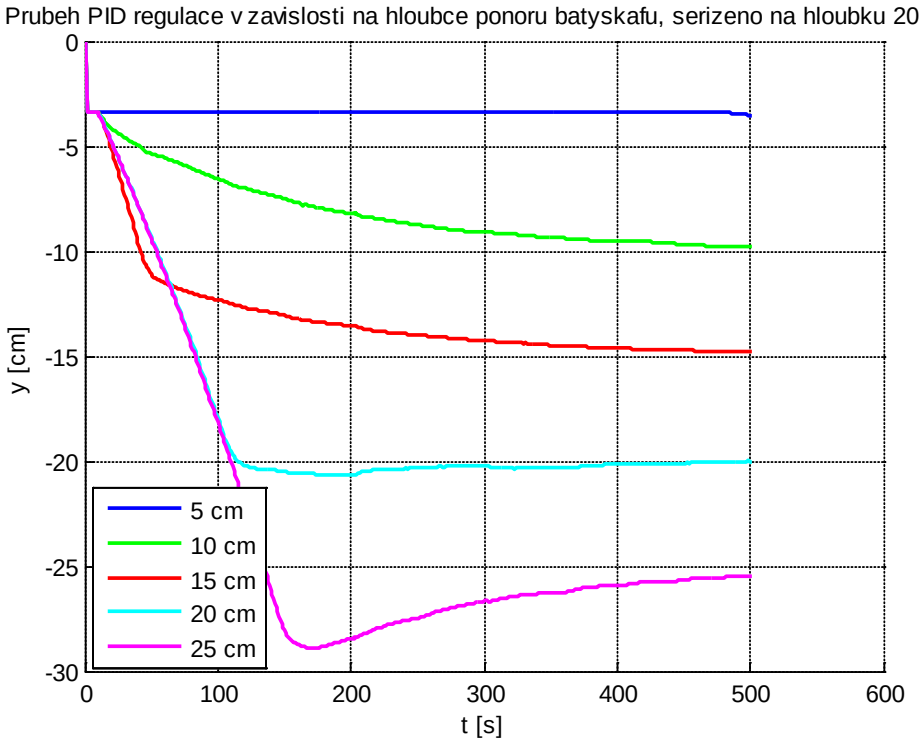


Obr. 4 – Odezva soustavy batyskafu regulované PID regulátorem na žádanou hodnotu hloubky při různých velikostech změn žádané veličiny

Regulátor PID byl postupně seřízen na hloubku 10 a 20 cm a pro každé seřízení byly odsimulovány odezvy na různé žádané hloubky (vždy z výchozí pozice horní polohy 3,3 cm) – postupně 5, 10, 15, 20 a 25 cm. Podle očekávání se prokázalo, že PID regulátor seřízený na jednu konkrétní hloubku nezvládne uregulovat batyskaf v hloubce jiné. PID regulátor byl seřízen pomocí metody Ziegler-Nicholse a dále pak experimentálně doladěn pro funkčnost s reálnou soustavou.



Obr. 5 – Průběhy PID regulace v závislosti na hloubce ponoru batyskafu - seřízení PID regulátru na hloubku 20 cm (p=5, i=0.2, d=3)



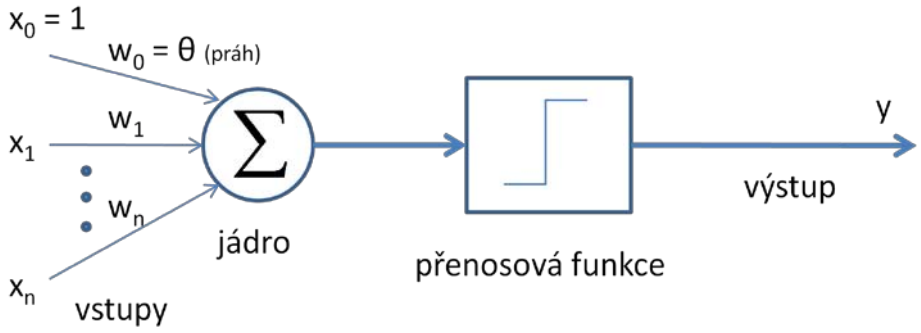
Obr. 6 – Průběhy PID regulace v závislosti na hloubce ponoru batyskafu – seřízení PID regulátru na hloubku 20 cm (p=5, i=0.03, d=3)

Na Obr. 5 jsou průběhy regulace PID regulátoru seřízeného na hodnotu hloubky 10 cm. Je patrné, že s hloubkou, na kterou je PID regulátor seřízen, si poradí poměrně dobře. Mnohem větší problémy nastanou, když má regulátor dobře regulovat dosažení hloubek jiných, než na které je seřízen. Větší hloubky než 10 cm zvládne batyskaf s takto seřízeným PID regulátorem jen s velkými překmity a nižší hloubky než 10 cm zvládne regulátor dosáhnout za velice dlouhou dobu – je pomalý.

Průběhy regulace PID regulátoru seřízeného na hloubku 20 cm jsou na Obr. 6. Zde je opět vidět, že seřízený PID regulátor dostane batyskaf s dobrou regulací na hodnotu, na kterou je vyladěn a naopak ostatní hodnoty batyskaf s tímto regulátorem dosahuje velice špatně. Také zde platí, že dosažení vyšších hloubek než, na které je regulátor seřízen, s sebou přináší velké překmity a dosažení nižších hloubek zase trvá velice dlouho.

3.2 Nelineární neuronové jednotky s nelinearitami vyšších řádů

Jednotky vyšších řádů (HONNU) se od klasických neuronových jednotek liší aktivační funkcí [1], tj. agregační funkcí [9] neurálních vstupů. Zatímco aktivační funkce klasického perceptronu je lineární funkcí, tak u HONNU jde vždy o funkci nelineární.

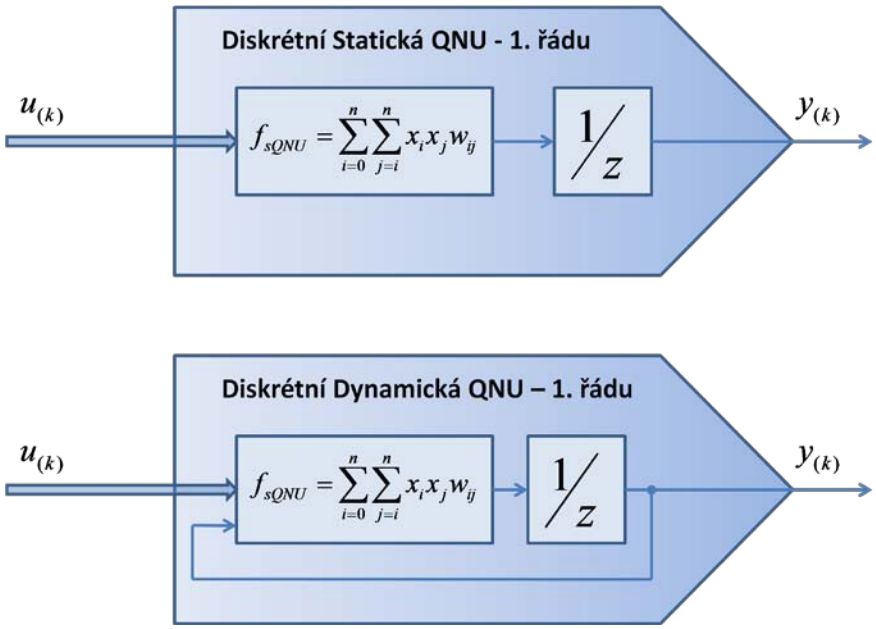


Obr. 7 – Schéma jednoduchého Perceptronu

$$f_{perceptronu} = \sum_{i=0}^{n} x_i w_i \tag{2}$$

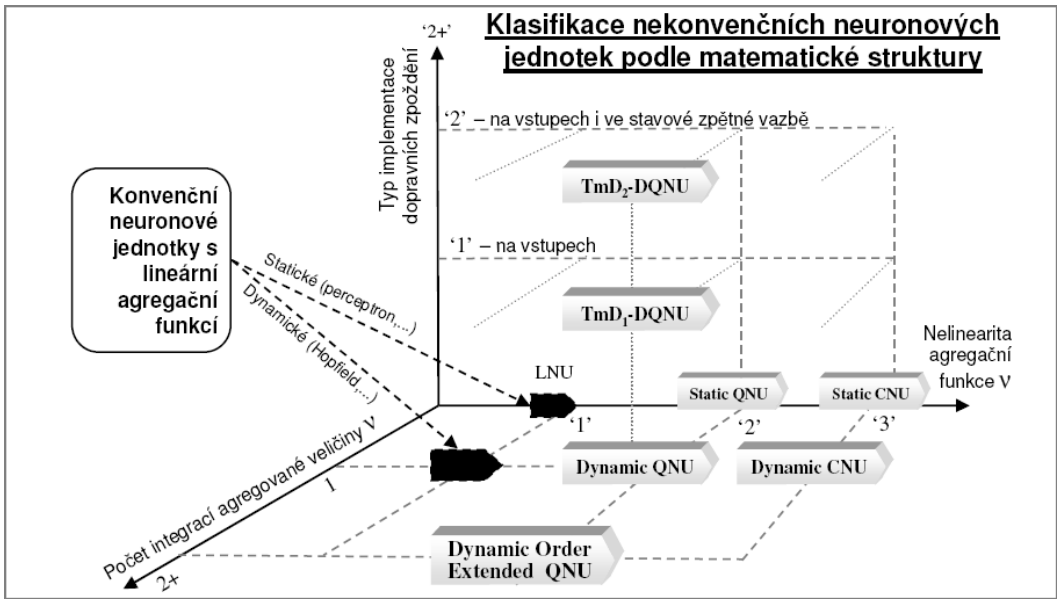
$$f_{HONNU-QNU} = \sum_{i=0}^{n}\sum_{j=i}^{n} x_i\, x_j w_{ij} \;;\; f_{HONNU-CNU} = \sum_{i=0}^{n}\sum_{j=i}^{n}\sum_{k=j}^{n} x_i x_j x_k w_{ij} \tag{3}$$

Neuronové jednotky HONNU lze rozdělit na jednotky statické a dynamické. Statická HONNU neobsahuje žádnou zpětnou vazbu zavedenou z výstupu a reaguje tedy vždy jen na aktuální stav systému. Naproti tomu dynamická HONNU obsahuje minimálně jednu zpětnou vazbu. Počet zpětných vazeb je dán řádem jednotky. Schéma statické a dynamické QNU je uvedeno na Obr. 8. Podrobné rozdělení nekonvenčních neuronových jednotek lze nalézt v literatuře [4][9].



Obr. 8 – Schéma diskrétní statické a dynamické QNU pro aproximaci soustav 1. řádu

Základními parametry neuronových jednotek jsou jednotlivé neurální váhy w_{ij} (myšleno hlavně jejich počáteční podmínky) a parametr rychlosti učení μ. Tyto parametry se většinou v praxi volí experimentálně (empiricky). Špatně zvolené počáteční podmínky jednotlivých neurálních vah můžou vést (a také při simulacích vedly [1][12]) k rozkmitání systému. Důvodem je volba počátečních podmínek v nestabilní oblasti. Na parametru rychlosti učení μ závisí nejen rychlost, s jakou se neuronová jednotka učí, ale i jak dobře se naučí. Stejně tak tento parametr je nutno volit s rozvahou a s jistou dávkou experimentování, protože například příliš vysoká hodnota může systém rozkmitat a naopak příliš nízká hodnota způsobí, že se neuronová jednotka učí tak pomalu, že je prakticky nepoužitelná. Důležitým parametrem u diskrétních neuronových jednotek je také perioda vzorkování. I tu je nutno volit s rozvahou.



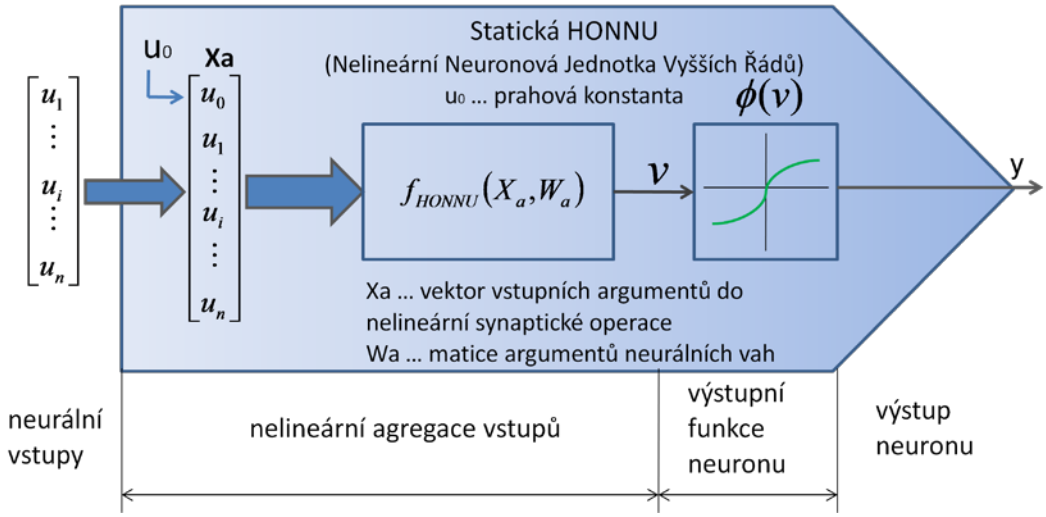
Obr. 9 – Klasifikace nekonvenčních neuronových jednotek (převzato z [9])

Rozdělení neuronových jednotek s dynamikou 1. Řádu		Typ agregace v			
		Lineární	Kvadratická	Kubická	Polynomy stupně>3
Typ implementace adaptabilních dopravních zpoždění	Bez zpoždění	DLNU	DQNU	DCNU	$DHONNU_4$
	Pouze na vstupu	TmD_1-DLNU	TmD_1-DQNU	TmD_1-DCNU	TmD_1-$DHONNU_4$
	Na vstupu i ve zpětné vazbě	TmD_2-DLNU	TmD_2-DQNU	TmD_2-DCNU	TmD_2-$DHONNU_4$

Tab. 1 – Rozdělení neuronových jednotek s dynamikou 1. řádu (převzato a upraveno z [3][9])

3.3 Statické HONNU

Základem každé neuronové jednotky je její agregační funkce, ve které dochází k vážení jednotlivých vstupů podle důležitosti přiřazených vah a následné sumaci podle funkce neuronu. Hlavní funkce může být předepsaná lineární nebo nelineární rovnicí. Často se jednotlivé části rozdělují na vstupní sumační funkci, samotnou agregační funkci neuronové jednotky a na výstupní nelinearitu neuronové jednotky $\phi()$ - označovanou jako přenosová funkce [1]. Příklady agregační funkce neuronové jednotky HONNU jsou uvedeny v (3).



Obr. 10 – Schéma statické HONNU (převzato a upraveno z [2])

Protože statické HONNU neovlivňuje jejich vnitřní dynamika, je jejich konvergence stabilnější a rychlejší. Naopak zase nedokáží tak citlivě aproximovat složitější dynamiku reálného systému. Statické HONNU se dobře uplatní k nalezení počátečních hodnot neurálních vah pro dynamické HONNU, [2].

3.3.1 Trénování statických HONNU – Batch-Training

Při učení neuronové jednotky podstatně záleží na vstupním signálu. Je-li signál nevhodný, tj. například obsahuje skokové změny (skoky), tak může dojít k rozhození neuronových vah a destabilizaci neuronové jednotky. V tomto případě lze použít metodu Batch-Training (BT). Jde o metodu, kdy se data neučí postupně, ale v dávkách. A to buď v jedné dávce najednou nebo v několika dávkách.

3.3.2 Levenberg-Marquardt algoritmus (LM)

Batch-Trainingová metoda vhodná pro potřeby, které se zde řeší, je například algoritmus Levenberg-Marquardt. Tento algoritmus je variantou Gauss-Newtonovy optimalizační metody. Byl navržen pro dosažení rychlosti učení druhého řádu [14].

$$x_{(k+1)} = x_{(k)} - [\boldsymbol{J}^T\boldsymbol{J} + \mu\boldsymbol{I}]^{-1}\boldsymbol{J}^T\boldsymbol{e}_{(\boldsymbol{k})} \qquad (4)$$

Podle Levenberg-Marquardtova algoritmu bude mít rovnice (4) pro výpočet změny neurálních vah v každém kroku tvar podle (5).

$$\Delta \boldsymbol{w}_{ij} = [\boldsymbol{J}^T\boldsymbol{J} + \mu\boldsymbol{I}]^{-1}\boldsymbol{J}^T\boldsymbol{e} \tag{5}$$

Matice $\boldsymbol{J}$ je Jacobiho matice (Jacobian) a obsahuje první derivace vektoru chyb podle jednotlivých neurálních vah. Je-li y_n výstupní funkce z neuronové jednotky a y_r je výstupní funkce z reálného systému, pak se vektor chyb vypočte podle (6).

$$e_{(k)} = y_{r\,(k)} - y_{n(k)} \tag{6}$$

Protože y_r není funkcí ani jedné z neurálních vah, můžeme rovnou provést parciální derivace pro agregační funkci neuronové jednotky y_n. Jacobiho matice $\boldsymbol{J}$ bude mít pak tvar (7).

$$J = \begin{bmatrix} \frac{\partial y_{n_1}}{\partial w_1} & \cdots & \frac{\partial y_{n_1}}{\partial w_n} & \frac{\partial y_{n_1}}{\partial w_0} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial y_{n_p}}{\partial w_1} & \cdots & \frac{\partial y_{n_p}}{\partial w_n} & \frac{\partial y_{n_p}}{\partial w_0} \end{bmatrix} \tag{7}$$

Jestliže je μ nulové, pak jde o klasickou Newtonovu metodu. Když je μ nenulové má to za důsledek klesající tendenci gradientu s malým krokem. Parametr μ se zmenšuje s každým neúspěšným krokem a zvyšuje se jen když by mohl v prozatímním kroku zvýšit výkonovou funkci. V důsledku toho se výkonová funkce vždy snižuje v každé iteraci algoritmu.

Mějme agregační funkci (8), která popisuje QNU (Quadratic Neural Unit). Jako vstup uvažujme vektor vstupních veličin $[x_0, x_1, x_2, x_3]$.

$$f_{QNU} = \sum_{i=0}^{n-1}\sum_{j=i}^{n-1} x_i x_j w_{ij} \tag{8}$$
$$, x_0 = 1, k \in R^1$$

Rozepíšeme-li agregační funkci v závislosti na počtu vstupů, dostaneme rovnici (9). Z agregační funkce (9) sestavíme Jacobiho matici (10) podle .

$$\begin{aligned} f_{QNU} = {} & w_{00}x_0^2 + w_{01}x_1 + w_{02}x_2 + w_{03}x_3 + w_{11}x_1^2 + w_{12}x_1x_2 \\ & + w_{13}x_1x_3 + w_{22}x_2^2 + w_{23}x_2x_3 + w_{33}x_3^2 \end{aligned} \tag{9}$$

$$J = \begin{bmatrix} \frac{\partial f_{qnu_1}}{\partial w_{00}} & \frac{\partial f_{qnu_1}}{\partial w_{01}} & \frac{\partial f_{qnu_1}}{\partial w_{02}} & \frac{\partial f_{qnu_1}}{\partial w_{03}} & \frac{\partial f_{qnu_1}}{\partial w_{11}} & \frac{\partial f_{qnu_1}}{\partial w_{12}} & \frac{\partial f_{qnu_1}}{\partial w_{13}} & \frac{\partial f_{qnu_1}}{\partial w_{22}} & \frac{\partial f_{qnu_1}}{\partial w_{23}} & \frac{\partial f_{qnu_1}}{\partial w_{33}} \\ \frac{\partial f_{qnu_2}}{\partial w_{00}} & \frac{\partial f_{qnu_2}}{\partial w_{01}} & \frac{\partial f_{qnu_2}}{\partial w_{02}} & \frac{\partial f_{qnu_2}}{\partial w_{03}} & \frac{\partial f_{qnu_2}}{\partial w_{11}} & \frac{\partial f_{qnu_2}}{\partial w_{12}} & \frac{\partial f_{qnu_2}}{\partial w_{13}} & \frac{\partial f_{qnu_2}}{\partial w_{22}} & \frac{\partial f_{qnu_2}}{\partial w_{23}} & \frac{\partial f_{qnu_2}}{\partial w_{33}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_{qnu_k}}{\partial w_{00}} & \frac{\partial f_{qnu_k}}{\partial w_{01}} & \frac{\partial f_{qnu_k}}{\partial w_{02}} & \frac{\partial f_{qnu_k}}{\partial w_{03}} & \frac{\partial f_{qnu_k}}{\partial w_{11}} & \frac{\partial f_{qnu_k}}{\partial w_{12}} & \frac{\partial f_{qnu_k}}{\partial w_{13}} & \frac{\partial f_{qnu_k}}{\partial w_{22}} & \frac{\partial f_{qnu_k}}{\partial w_{23}} & \frac{\partial f_{qnu_k}}{\partial w_{33}} \end{bmatrix} \quad (10)$$

Jedná se o klasickou Jacobiho matici sestavenou z parciálních derivací chybové funkce podle jednotlivých neurálních vah. Zde již uvažujeme rovnici (6), kde $yn = f_{qnu}$, protože y_r (výstup z reálné soustavy) není závislá na žádné z neurálních vah. Parametr k určuje hodnotu diskrétního kroku. V rovnici (11) je jeden řádek matice J pro $k = 1$. Chybový vektor je uveden v rovnici (12).

$$J = \begin{bmatrix} 1 & x_{0_{(k)}}x_{1_{(k)}} & x_{0_{(k)}}x_{2_{(k)}} & x_{0_{(k)}}x_{3_{(k)}} & x_{1_{(k)}}^2 & x_{1_{(k)}}x_{2_{(k)}} & x_{1_{(k)}}x_{3_{(k)}} & x_{2_{(k)}}^2 & x_{2_{(k)}}x_{3_{(k)}} & x_{3_{(k)}}^2 \end{bmatrix} \quad (11)$$

$$e = \begin{bmatrix} e_1 & e_2 & \cdots & e_k \end{bmatrix} \quad (12)$$

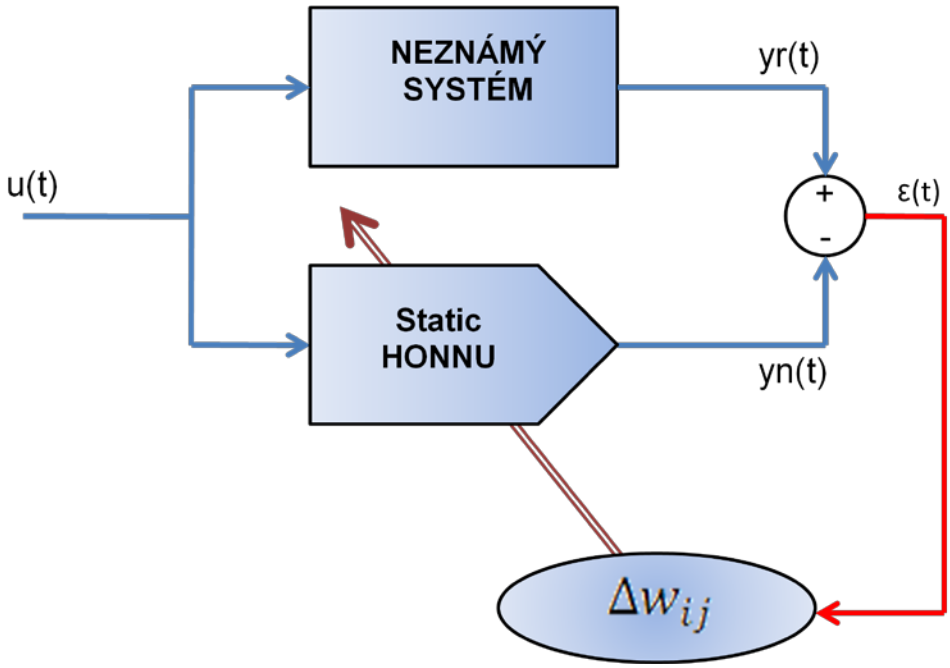
Nyní máme definováno vše potřebné k výpočtu změn neurálních vah pro každý další krok. Dosadíme do rovnice (4) a dostaneme výsledný vztah (13).

$$\Delta \boldsymbol{w}_{ij} = \left[\begin{bmatrix} \frac{\partial y_{n_1}}{\partial w_1} & \cdots & \frac{\partial y_{n_1}}{\partial w_n} & \frac{\partial y_{n_1}}{\partial w_0} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial y_{n_p}}{\partial w_1} & \cdots & \frac{\partial y_{n_p}}{\partial w_n} & \frac{\partial y_{n_p}}{\partial w_0} \end{bmatrix}^T \begin{bmatrix} \frac{\partial y_{n_1}}{\partial w_1} & \cdots & \frac{\partial y_{n_1}}{\partial w_n} & \frac{\partial y_{n_1}}{\partial w_0} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial y_{n_p}}{\partial w_1} & \cdots & \frac{\partial y_{n_p}}{\partial w_n} & \frac{\partial y_{n_p}}{\partial w_0} \end{bmatrix} + \mu \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} \right] \cdot \begin{bmatrix} \frac{\partial y_{n_1}}{\partial w_1} & \cdots & \frac{\partial y_{n_1}}{\partial w_n} & \frac{\partial y_{n_1}}{\partial w_0} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial y_{n_p}}{\partial w_1} & \cdots & \frac{\partial y_{n_p}}{\partial w_n} & \frac{\partial y_{n_p}}{\partial w_0} \end{bmatrix}^T \begin{bmatrix} e_1 \\ \vdots \\ e_k \end{bmatrix} \quad (13)$$

$$, kde\ \Delta \boldsymbol{w}_{ij} = [\Delta w_{00}\ \Delta w_{01}\ \ldots\ \Delta w_{nm}]$$

3.3.3 Adaptace statických HONNU

Pro identifikaci parametrů soustavy se dá snadno použít algoritmus Back-Propagation (BP), který u ď statickou HONNU proti gradientu chyby [2][8]. Tento algoritmus je snadno aplikovatelný pro mnoho různých tříd umělých neuronových systémů. Schéma učícího procesu je ukázáno na Obr. 11.



Obr. 11 – Algoritmus učení Back-Propagation

Během gradientového učení podle algoritmu Back-Propagation se neurální váhy učí proti směru gradientu výkonnostního indexu J (15). Změna přírůstku jednotlivých neurálních vah je uvedena v (14).

$$\Delta w_{ij} = -\mu \frac{\delta J}{\delta w_{ij}} \tag{14}$$

$$J = \frac{1}{2}\varepsilon^2 \tag{15}$$

, kde Δw_{ij} je přírůstek neurálních vah, J je výkonový index, μ je rychlost učení a ε je chyba mezi výstupem z neuronu a požadovaným výstupem.

Následující hodnota neurální váhy se vypočítá podle (16) [2][8].

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k)$$
$$\Delta w_{ij}(k) = -\frac{1}{2}\mu \frac{\delta \varepsilon^2}{\delta w_{ij}} = -\mu\varepsilon \frac{\delta(y_r - y_n)}{\delta w_{ij}} = \mu\varepsilon \frac{\delta f_{HONNU}(u_i, w_{ij})}{\delta w_{ij}} \tag{16}$$

3.4 Diskrétní Dynamické HONNU

Jak již bylo zmíněno v kapitole 3.3, dynamická neuronová jednotka se od statické liší zejména zavedením jedné nebo několika zpětných stavových vazeb. Počet zpětných vazeb zvyšuje schopnost neuronové jednotky aproximovat systémy vyšších řádů. Například neuronová jednotka s dynamikou 2. řádu by měla být velice dobře schopná aproximovat nelineární systémy 2. řádu.

Do agregační funkce neuronové jednotky potom vstupují, kromě vstupního vektoru, ještě vstupy od jednotlivých zpětných vazeb. Schéma diskrétní dynamické QNU s dynamikou 2. řádu je uvedeno na Obr. 12.

3.5 HONNU jako zpětnovazební stavový adaptabilní regulátor

Protože neuronové jednotky mají stabilní oblasti v blízkosti počátku souřadnicového systému (detailnější popis stabilních oblastí nelineárních systémů lze nalézt v [2] str. 30.) bylo pro dobrou schopnost učení a aproximace neuronové jednotky nutné vstupní i výstupní data normalizovat na hodnoty blízké stabilní oblasti, tudíž mezi -1 a 1. S nenormalizovanými daty byly pro neuronovou jednotku náročně zvladatelné výchylky měnící se ve velkém rozsahu hodnot a projevilo se to právě špatnou aproximací dat a špatným učením se na taková data. Z tohoto důvodu bylo nutné vždy vstupní data do neuronových jednotek normalizovat a výstupní denormalizovat.

Dalším problémem byly příliš velké skoky u vstupní veličiny *u*, které vždy rozhodily konvergenci neurálních vah neuronových jednotek. Během experimentů se ukázalo, že si s těmito skoky neuronové jednotky do jisté míry poradí a že je dokáží minimalizovat.

Současný běh identifikace a Neuro-regulátoru najednou může vést ke špatné regulaci, protože soustava nebude dostatečně identifikována a neurální váhy nebudou zcela zkonvergované. Proto je lepším postupem soustavu nejdříve dobře zidentifikovat (to bylo provedeno cvičením diskrétní dynamické QNU po 1500 epoch) a výsledky identifikace použít pro nastavení Neuro-regulátoru.

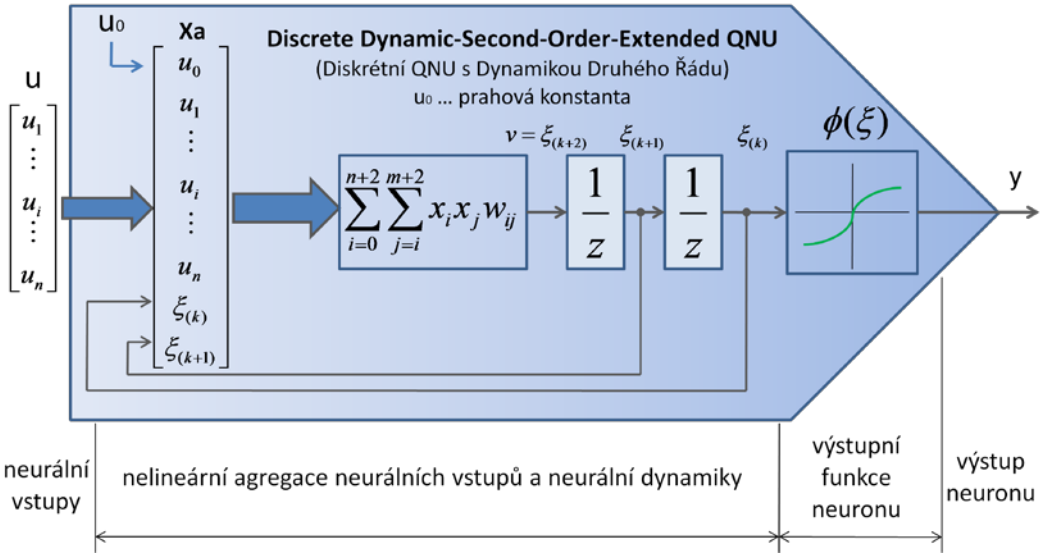
4 Řešení

4.1 Volba nelineárního modelu – QNU

Protože soustava batyskafu je nelineární soustavou 2. řádu (1), měla by pro identifikaci a správnou aproximaci být plně dostačující kvadratická neuronová jednotka (QNU) s dynamikou 2. řádu. Pro představu a porovnání byla vyzkoušena také identifikace soustavy batyskafu pomocí lineární neuronové jednotky LNU nebo QNU s nižším řádem dynamiky, ale ukázalo se, že nejlépe lze systém batyskafu opravdu aproximovat zmíněnou QNU s dynamikou 2. řádu. Zvolit vyšší řád dynamiky neuronové jednotky nebo vyšší řád agregační funkce sice možné je, ale nelineární model takové neuronové jednotky by byl zbytečně složitější, výpočtově náročnější a hlavně by pravděpodobně aproximoval soustavu mnohem hůře.

Tyto předpoklady a poznatky navíc potvrzuje metoda hledání falešných blízkých sousedů (FNN), která je detailně popsána v následující kapitole 4.1.1. Metoda FNN se používá k určení vnořené dimenze, při které se počet falešných blízkých sousedů sníží pod stanovenou hranici. Takto nalezená nebo určená vnořená dimenze pak odpovídá vhodnému počtu vstupních hodnot (vektorů) do neuronové jednotky a podle toho je možné odhadnout také vhodný řád dynamiky neuronové jednotky.

Z těchto důvodů byla zvolena pro identifikaci soustavy batyskafu diskrétní dynamická QNU s dynamikou 2. řádu (DDSOE-QNU [2][3][4][9]). Základní schéma je na Obr. 12.



Obr. 12 – Základní schéma diskrétní dynamické QNU s dynamikou 2. řádu (převzato a upraveno z [2])

V konkrétním případě soustavy batyskafu vstupuje do systému jedna vstupní hodnota, která je reprezentována vektorem $\boldsymbol{u}$, který obsahuje diskrétní hodnoty v po sobě jdoucích krocích. Jedná se o hodnoty natočení serva, které svým natočením reguluje tlak nad hladinou (dodávaný kompresorem). Dalšími vstupními hodnotami jsou zavedené dynamické zpětné vazby $y_{(k)}$ a $y_{(k+1)}$, a prahová konstanta u_0. Máme tedy vstupní vektor se čtyřmi hodnotami (17). Tento vstupní vektor vstupuje do agregační funkce (18) neuronové jednotky, takže po rozepsání agregační funkce v závislosti na počtu vstupů bude vnitřní funkce neuronové jednotky obsahovat součty součinů jednotlivých neurálních vah s konkrétními vstupy. Rovnice (19) zobrazuje rozepsanou agregační funkci.

$$\boldsymbol{x_a} = \left[x_0\ u_{(k)}\ y_{(k)}\ y_{(k+1)}\right]^T \tag{17}$$

$$f_{QNU} = \sum_{i=0}^{n-1}\sum_{j=i}^{n-1} x_i x_j w_{ij} \tag{18}$$

$$, kde\, n = 4\ (počet\ vstupních\ hodnot)$$

$$\begin{aligned} f_{QNU} = {} & w_{00} + w_{01}u_{(k)} + w_{02}y_{(k)} + w_{03}y_{(k+1)} + w_{11}u_{(k)}^2 \\ & + w_{12}u_{(k)}y_{(k)} + w_{13}u_{(k)}y_{(k+1)} + w_{22}y_{(k)}^2 \\ & + w_{23}y_{(k)}y_{(k+1)} + w_{33}y_{(k+1)}^2 \end{aligned} \tag{19}$$

Matice neurálních vah je uvedena v (20). Kompletní obecný maticový zápis agregační funkce je uveden v rovnici (21) a konkrétní maticový zápis pro diskrétní dynamickou QNU je uveden v (22).

$$W = \begin{bmatrix} w_{00} & w_{01} & w_{02} & w_{03} \\ 0 & w_{11} & w_{12} & w_{13} \\ 0 & 0 & w_{22} & w_{23} \\ 0 & 0 & 0 & w_{33} \end{bmatrix} \tag{20}$$

$$f_{QNU} = \boldsymbol{x_a} \cdot \boldsymbol{W} \cdot \boldsymbol{x_a^T} \tag{21}$$

$$f_{QNU} = \begin{bmatrix} u_0 & u_{(k)} & y_{(k)} & y_{(k+1)} \end{bmatrix} \cdot \begin{bmatrix} w_{00} & w_{01} & w_{02} & w_{03} \\ 0 & w_{11} & w_{12} & w_{13} \\ 0 & 0 & w_{22} & w_{23} \\ 0 & 0 & 0 & w_{33} \end{bmatrix} \cdot \begin{bmatrix} u_0 \\ u_{(k)} \\ y_{(k)} \\ y_{(k+1)} \end{bmatrix} \tag{22}$$

Výpočet přírůstků jednotlivých neurálních vah je v maticovém zápisu, pro konkrétní diskrétní dynamickou QNU, uveden v (23).

$$\Delta W_{ij} = \mu \cdot e \cdot \begin{bmatrix} \frac{\partial f_{QNU}}{\partial w_{00}} & \frac{\partial f_{QNU}}{\partial w_{01}} & \frac{\partial f_{QNU}}{\partial w_{02}} & \frac{\partial f_{QNU}}{\partial w_{03}} \\ 0 & \frac{\partial f_{QNU}}{\partial w_{11}} & \frac{\partial f_{QNU}}{\partial w_{12}} & \frac{\partial f_{QNU}}{\partial w_{13}} \\ 0 & 0 & \frac{\partial f_{QNU}}{\partial w_{22}} & \frac{\partial f_{QNU}}{\partial w_{23}} \\ 0 & 0 & 0 & \frac{\partial f_{QNU}}{\partial w_{33}} \end{bmatrix} = \mu \cdot e \cdot \begin{bmatrix} 1 & u_{(k)} & y_{(k)} & y_{(k+1)} \\ 0 & u_{(k)}^{\,2} & u_{(k)} y_{(k)} & u_{(k)} y_{(k+1)} \\ 0 & 0 & y_{(k)}^{\,2} & y_{(k)} y_{(k+1)} \\ 0 & 0 & 0 & y_{(k+1)}^{\,2} \end{bmatrix} \tag{23}$$

4.1.1 False Nearest Neighbors - FNN

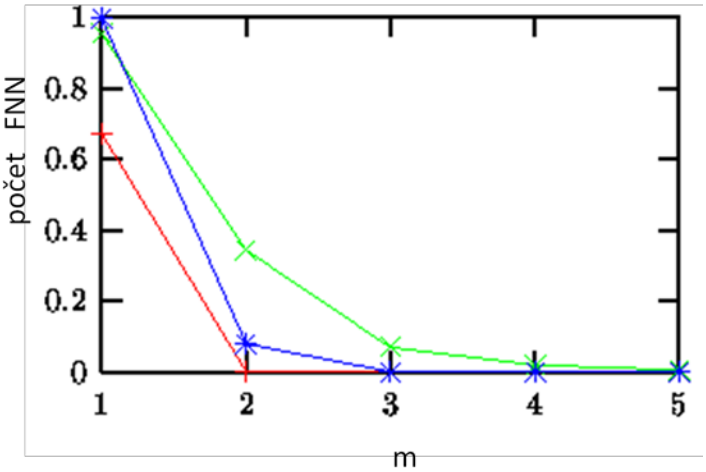
Aby se neuronová jednotka mohla dobře učit, musí mít pro každý jasně definovaný vstup také jasně definovaný jediný výstup. Jednoduše řečeno, bude-li se na vstupu opakovat nějaká konkrétní hodnota, musí této hodnotě odpovídat výstupní hodnota a pro stejné opakující se hodnoty na vstupu by měla být také stejná.

Z toho důvodu byla formulována metoda False Nearest Neighbors (FNN), metoda Falešných Blízkých sousedů [10][11]. Algoritmus FNN hledá minimální vnořenou dimenzi při které klesne počet falešných sousedů pod stanovenou mez. Dostatečná mez může být taková, že počet bodů, které jsou označeny jako falešní sousedé je nulový nebo dostatečně malý.

Algoritmus FNN [10][11] funguje následovně: pro každý bod R_i v časové řadě se hledá blízký bod R_j v m-rozměrném prostoru. Spočte se jejich vzdálenost $|R_i - R_j|$ a opakovanými iteracemi se počítá prahová hodnota (24).

$$R_d = \frac{|R_{i+1} - R_{j+1}|}{|R_i - R_j|} \tag{24}$$

Jestliže hodnota R_d překročí heuristicky stanovený práh R_t, pak je tento bod označen jako falešný blízký soused – FNN. Na Obr. 13 je zobrazena závislost snižujícího se počtu falešných sousedů na zvyšující se vnořené dimenzi *m*.



Obr. 13 – FNN v závislosti na vnořené dimenzi *m*, modrá – Lorenzova řada (Lorenz series [11]), červená – Hénonova řada, zelená – Hénonova řada s 10% šumu, (převzato a upraveno z [11])

V našem konkrétním případě, s použitím Dynamické QNU pro aproximaci soustav 2. řádu Obr. 12, vstupují do QNU tři parametry. Jednak přímo vstupní hodnota $u_{(k)}$, která reprezentuje natočení serva ventilu regulujícího tlak nad hladinou, a spolu s ní dvě

zpětné vazby $y_{(k)}$ a $y_{(k+1)}$. Vstupní data musí být normalizována (zde mezi hodnoty 0 až 100), abychom měli jistotu, že ve výpočtech nepřeváží nedůležité složky. Je tedy nutné ověřit, zda jsou tyto vstupy pro QNU dostatečné a hlavně jestli je dostatečný jejich počet. Jak bylo vysvětleno výše, může se stát, že stávající vstupy a jejich počet nebudou dobře a jednoznačně charakterizovat soustavu a potom bychom museli přidat další vstup (například $y_{(k+2)}$) a tím zvýšit dimenzionalitu ze 3 na 4.

$$\begin{array}{c} \\ k=1 \\ k=2 \\ \vdots \\ k=n \end{array}\begin{bmatrix} u_{(k)} & y_{(k)} & y_{(k+1)} \\ u_{(k+1)} & y_{(k+1)} & y_{(k+2)} \\ \vdots & \vdots & \vdots \\ u_{(k+n-2)} & y_{(k+n-2)} & y_{(k+n-1)} \end{bmatrix} \Rightarrow \begin{bmatrix} y_{(k+2)} \\ y_{(k+3)} \\ \vdots \\ y_{(k+n)} \end{bmatrix} \downarrow k$$

vstupy *výstup*

Obr. 14 – Hledání FNN (matice vstupních vektorů a výstupní vektor QNU)

Nejprve je třeba určit vzdálenost mezi jednotlivými řádkovými vektory matice vstupů $\boldsymbol{X}$ viz Obr. 14. Ta se vypočte podle vzorce (25). Spočte se vzdálenost každého řádkového vektoru s každým.

$$dist_{i,j} = \sqrt{\left(O_i - O_j\right)^2 + \left(P_i - P_j\right)^2 + \left(Q_i - Q_j\right)^2} \tag{25}$$

, kde O … je první sloupec,

P … je druhý sloupec,

Q … je třetí sloupec vstupní matice $\boldsymbol{X}$,

i,j … jsou indexy příslušných řádků vstupní matice $\boldsymbol{X}$.

Z vypočtených vzdáleností *i*-tého a *j*-tého řádku sestavíme matici vzdáleností $\boldsymbol{D}$, která bude mít velikost $i \times j$ (26). Matice $\boldsymbol{D}$ má na diagonále samé nuly, protože vzdálenost každého vektoru od sebe sama je nulová. Hodnoty umístěné zrcadlově pod a nad diagonálou budou stejné, protože například $dist_{1,2} = dist_{2,1}$.

$$D = \begin{bmatrix} dist_{1,1} & dist_{1,2} & \cdots & dist_{1,j} \\ dist_{2,1} & dist_{2,2} & \cdots & dist_{2,j} \\ \vdots & \vdots & \cdots & \vdots \\ dist_{i,1} & dist_{i,2} & \cdots & dist_{i,j} \end{bmatrix} \tag{26}$$

Algoritmus funguje tak, že nalezne v matici $\boldsymbol{D}$ souřadnice s minimální hodnotou vzdálenosti vektorů. Tyto souřadnice přesně odpovídají umístění vektorů v matici

vstupů $\boldsymbol{X}$ (řádkové vektory na pozici $k = i, k = j$) a znamená to, že tyto dva (nebo více) vektory jsou si nejblíže. Dalším krokem je volba vstupní tolerance Tol_{in}, která bude určovat, které vektory považujeme za blízké a které už ne. Hodnota vstupní tolerance se připočte k nalezené minimální hodnotě vzdáleností. Tím získáme skupinu řádkových vektorů matice $\boldsymbol{X}$, o kterých můžeme říci, že jsou si dostatečně blízké.

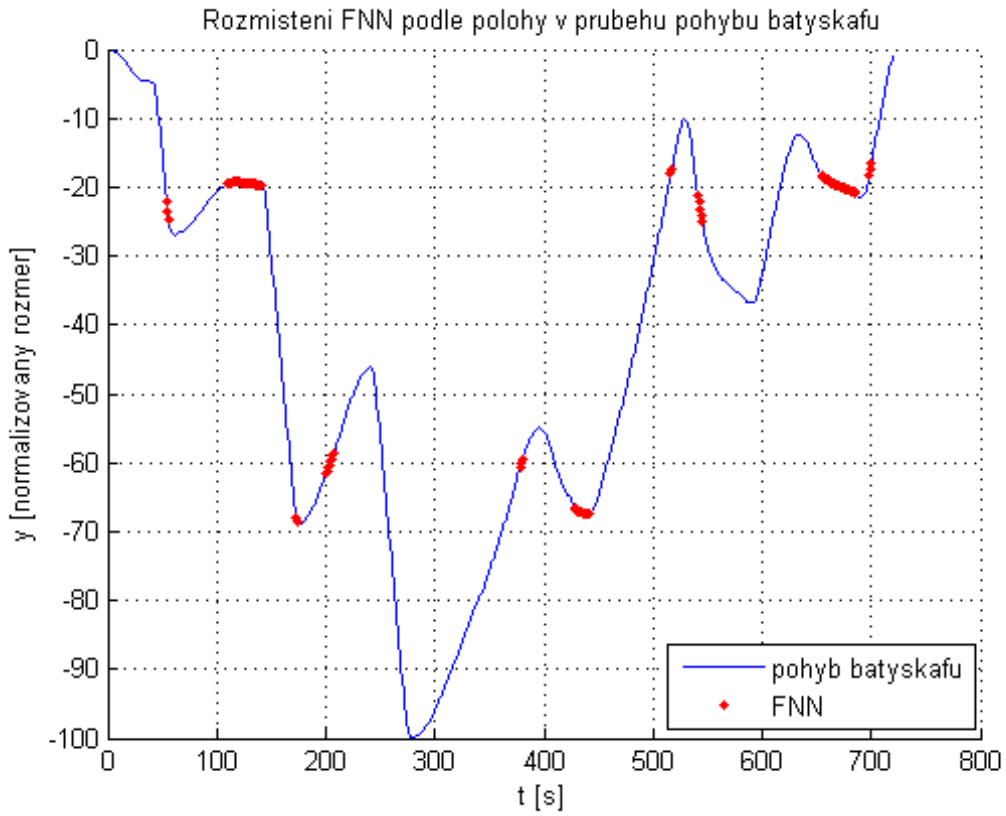
Hodnotu vstupní tolerance budeme volit podle vztahu (27), kde *x* označuje prvek vstupního vektoru, dolní index *k* označuje pozici prvku a horní index označuje číslo vstupního vektoru. V našem případě jsou vstupní vektory 3 ($u_{(k)}, y_{(k)}, y_{(k+1)}$). Jde tedy o procentuální vyjádření celkové tolerance odchylky vzdálenosti daných vektorů. Zvolíme-li například procentuální vstupní toleranci $Tol_{in} = 1$, říkáme tím, že maximální tolerance je 1%, ale to znamená, že buď všechny členy mohou být rovné hodnotě přibližně 0,3334 nebo jeden bude rovný nule a ostatní dva rovné přibližně 0,6667, a nebo dva členy budou nulové a jen jeden rovný 1.

$$Tol_{in} = \sqrt{\left(x_{1_{(k)}} - x_{1_{(k+1)}}\right)^2 + \left(x_{2_{(k)}} - x_{2_{(k+1)}}\right)^2 + \left(x_{3_{(k)}} - x_{1_{(k+1)}}\right)^2} \qquad (27)$$

V našem případě budeme volit právě $Tol_{in} = 1\%$, protože taková odchylka v sobě skryje nepřesnost čidel soustavy batyskafu a ještě je to dostatečně nízká a tím i přísně zvolená hodnota. Přísné kriterium pro hledání FNN zaručuje kvalitní výsledky.

Každý vektor z identifikovaných blízkých vektorů ukazuje na jeden prvek ve výstupním vektoru $y_{(k+2)}$ – řádkovému vektoru v matici vstupů na souřadnici například $k = 100$ odpovídá hodnota ve výstupním vektoru také na souřadnici $k = 100$. V ideálním případě by měly výstupní hodnoty, které patří vstupním blízkým vektorům, nabývat také podobných hodnot – být také blízko u sebe. Zde je také nutné zavést míru tolerance (označme jako výstupní tolerance - Tol_{out}), která určí, které výstupní hodnoty jsou ještě blízko u sebe a které už nejsou. Hodnoty, které budou mimo tuto výstupní toleranci nazveme falešnými blízkými sousedy (FNN).

Protože normalizované hodnoty se pohybují v rozmezí 0 až 100, můžeme dle zkušeností říci, že hodnota výstupní tolerance nastavená na 1 bude dostatečně přísná (odpovídá 1%).



Obr. 15 – Zobrazení počtu a polohy FNN v průběhu pohybu batyskafu (vstupní tolerance Tol_{in}=1%, výstupní tolerance Tol_{out}=1%)

Nastavíme-li vstupní toleranci Tol_{in}=1% a výstupní toleranci Tol_{out}=1% a spustíme algoritmus hledání FNN, získáme výstupní graf Obr. 15 a následující údaje:

- celkový počet vstupních vektorů = **718**
- počet výstupních hodnot, které překračují výstupní toleranci a je nutné je označit jako Falešné blízké sousedy – FNN = **104**
- procento počtu FNN z celkového počtu kombinací vstupních vektorů = **14,48%**

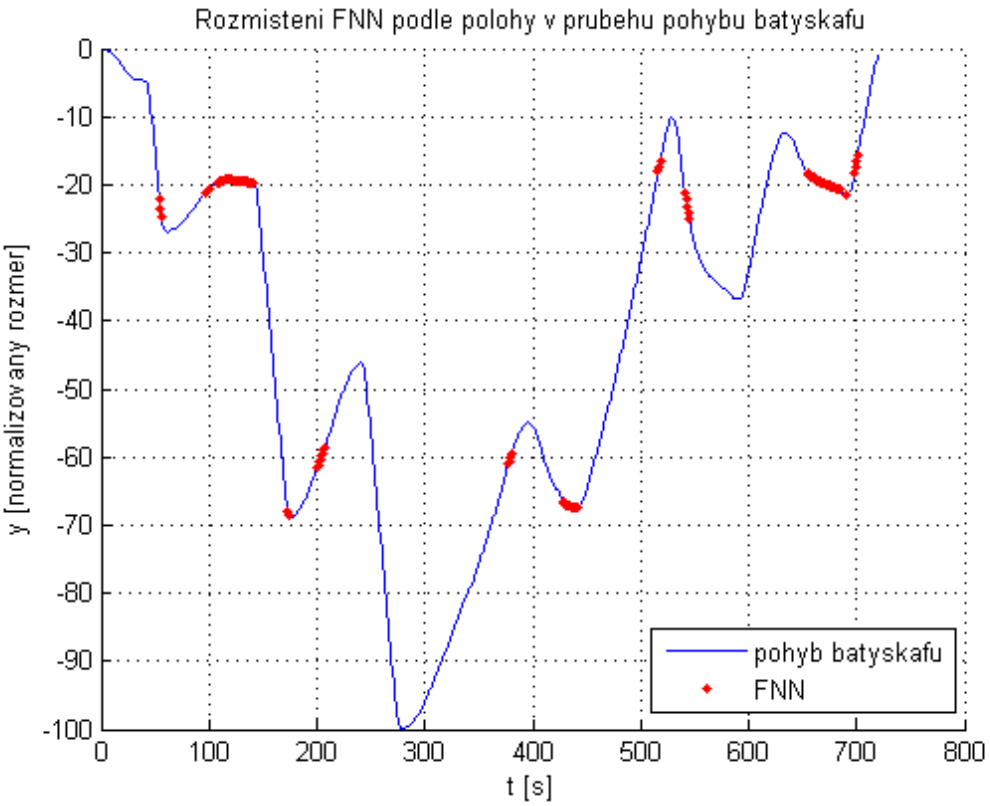
Z těchto výstupních dat vyplývá poznatek, že vstupní data do QNU jednoznačně definují 85,52% výstupních hodnot. Protože se zvýšením dimenzionality se již procento FNN nesníží (naopak se ještě trošku zvýší – viz níže) byl pro identifikaci soustavy diskrétní dynamickou QNU zvolen tento počet vstupních vektorů.

Dimenzionalita neuronu tedy nebyla dále zvýšena zavedením další stavové veličiny na vstup neuronu nebo znovu vstupní veličinou posunutou o jeden krok k. Pro porovnání zde uvedeme výsledky totožného algoritmu

s rozšířeným počtem vstupních vektorů a tím i vyšší vnořenou dimenzí. Vektor vstupů je tedy $[u_{(k)}\ y_{(k)}\ y_{(k+1)}\ y_{(k+2)}]$ a výstup je $y_{(k+3)}$.

Výsledné hodnoty pak (při shodně nastavených tolerancích) jsou:

- celkový počet kombinací vstupních vektorů = **718**
- počet výstupních hodnot, které překračují výstupní toleranci a je nutné je označit jako Falešné blízké sousedy – FNN = **115**
- procento počtu FNN z celkového počtu kombinací vstupních vektorů = **16,02%**



Obr. 16 – Počet a poloha FNN při vnořené dimenzi = 4

Je tedy vidět, že zvýšením dimenze (přidáním vstupu do neuronového modelu) se výsledky nijak radikálně nezměnily a procento FNN se i zvětšilo. Proto je původní počet vstupů a s tím i dimenze považována za dostačující pro účely této práce.

4.2 Identifikace batyskafu pomocí QNU

Statická identifikace reálného systému batyskafu byla provedena z naměřených dat. Data, uložená v jedné dávce, se použila pro Batch-Trainingovou (BT) metodu učení neuronové jednotky Levenberg-Marquardt (LM - kapitola 3.3.2).

Dynamická identifikace reálného systému batyskafu proběhla tak, že se neuronová jednotka připojila paralelně k reálnému systému. Dynamická identifikace využívá algoritmu učení Back-Propagation (BP). Vstupní signál, který vstupuje do reálného systému, je vstupem také pro neuronovou jednotku. Proces dynamické identifikace je znázorněn na Obr. 11.

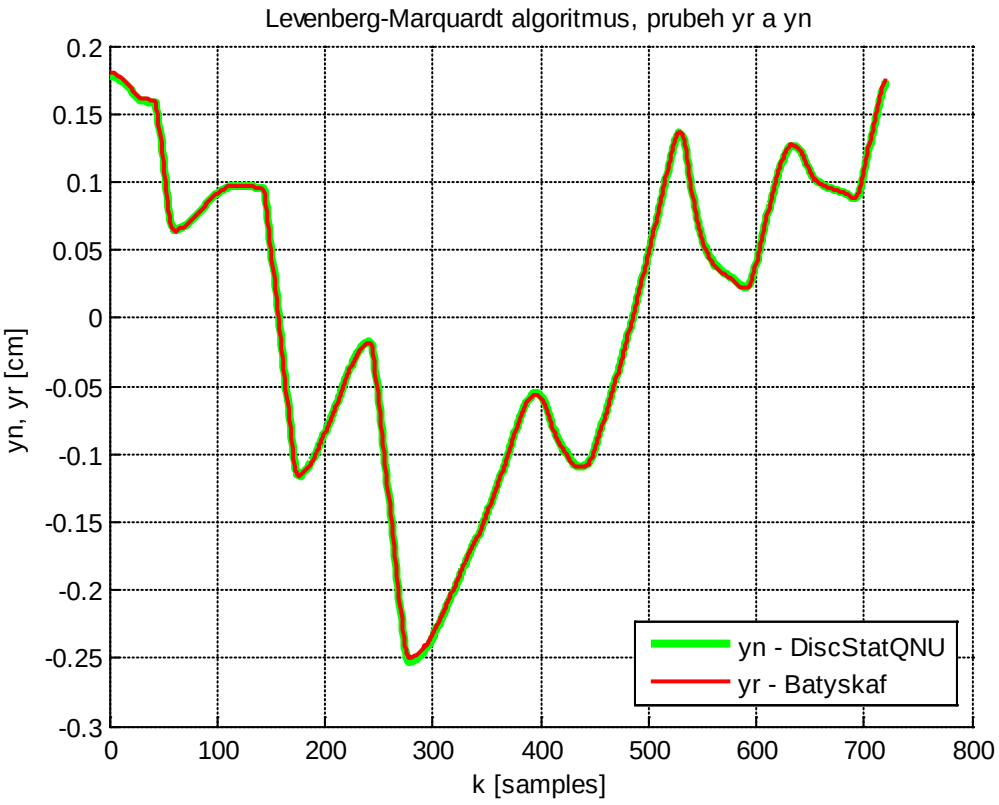
4.2.1 Batch-Training (statická QNU)

Batch-Training je, na rozdíl od kontinuálního učení, kdy se neuronová jednotka trénuje postupně v čase na jednotlivé vzorky, učením pro všechna data najednou – v jedné dávce (tedy metoda statická). Následují výsledky identifikace Batch-Trainingovou metodou Levenberg-Marquardt (LM), která byla detailně popsána v kapitole 3.3.1 a 3.3.2.

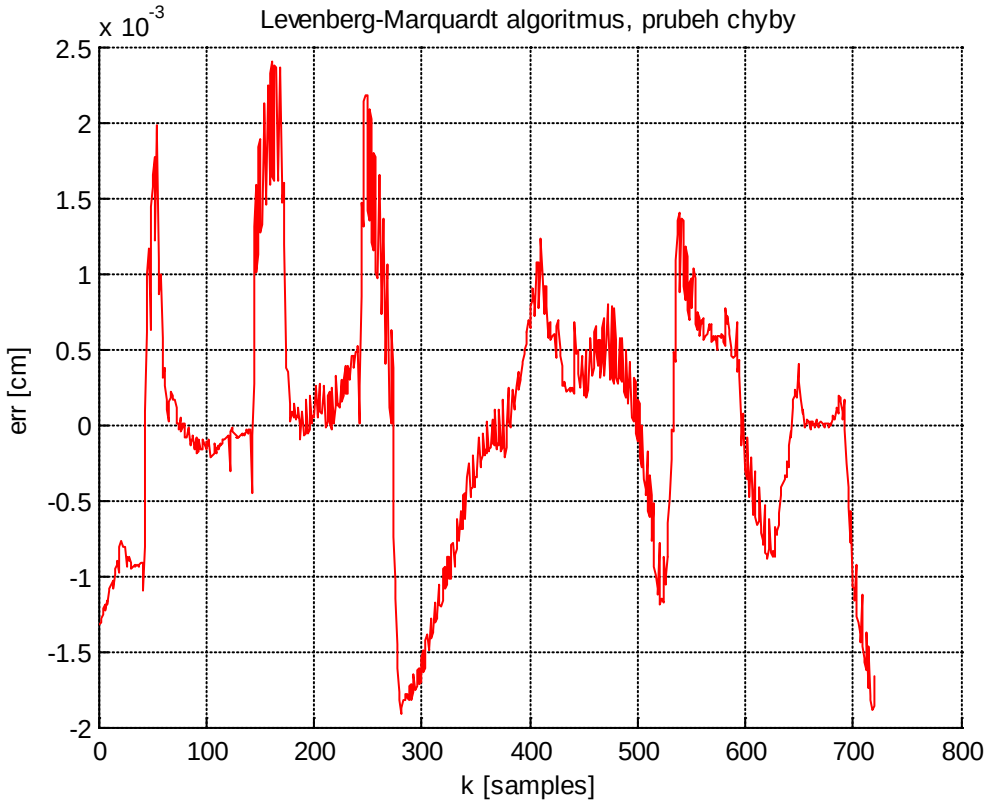
Statické metody většinou velice dobře konvergují a jsou velice efektivní. Vysoká stabilita metody umožňuje volbu vysoké rychlosti učení μ. Navíc se v každém kroku parametr učení mění v závislosti na úspěšné nebo neúspěšné iteraci. Při Batch-Trainingu jsou také potlačeny jinak nežádoucí skokové změny vstupního signálu na průběh konvergence neurálních vah. Z těchto důvodů probíhá učení neuronové jednotky velice rychle.

Simulační experimenty posléze naopak ukázaly, že Batch-Trainingová identifikace neposkytla očekávané dobré výchozí nastavení pro spuštění adaptabilního Neuro-regulátoru. Z tohoto důvodu nebyla použita metoda pro neuronové jednotky pracující v reálném čase.

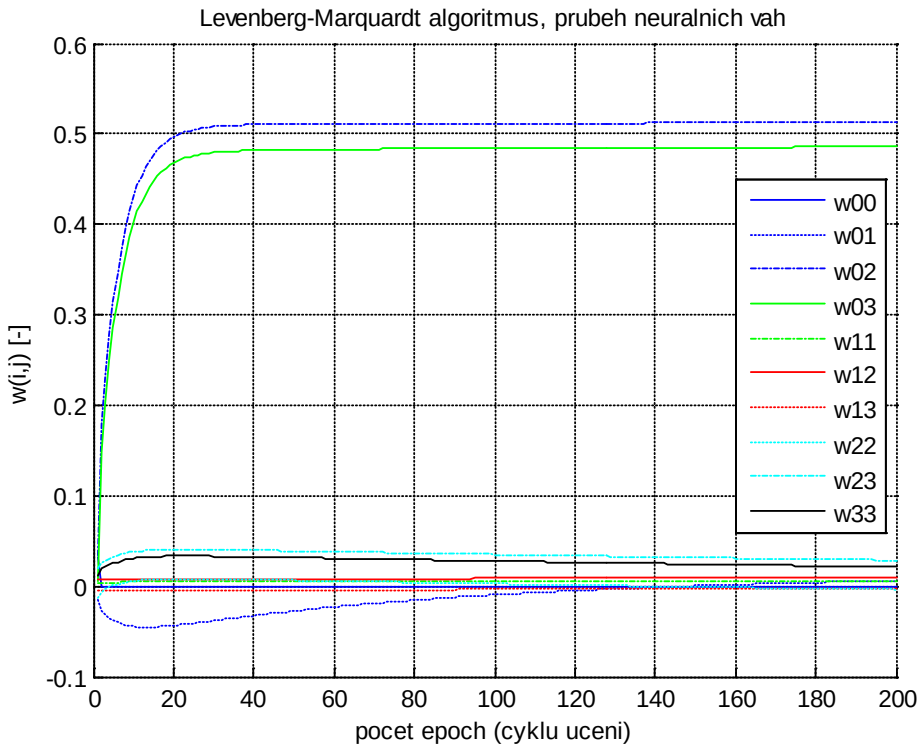
Zde budeme trénovat diskrétní statickou QNU, kde vstupem budou vektory $u_{(k)}, y_{(k)}, y_{(k+1)}$ a výstupem $y_{(k+2)}$.



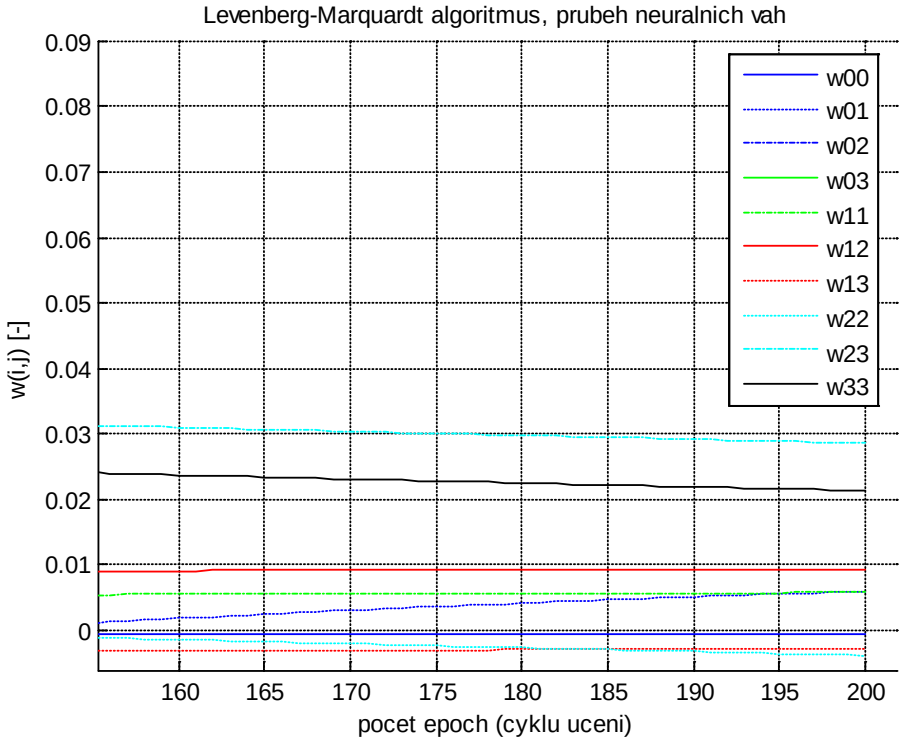
Obr. 17 – Levenberg-Marquardt algoritmus – průběh y_r a y_n



Obr. 18 - Levenberg-Marquardt algoritmus – průběh chyby



Obr. 19 - Levenberg-Marquardt algoritmus – průběh neurálních vah $w_{i,j}$ v závislosti na epochách (neurální váhy jednoznačně konvergují, identifikace batyskafu)



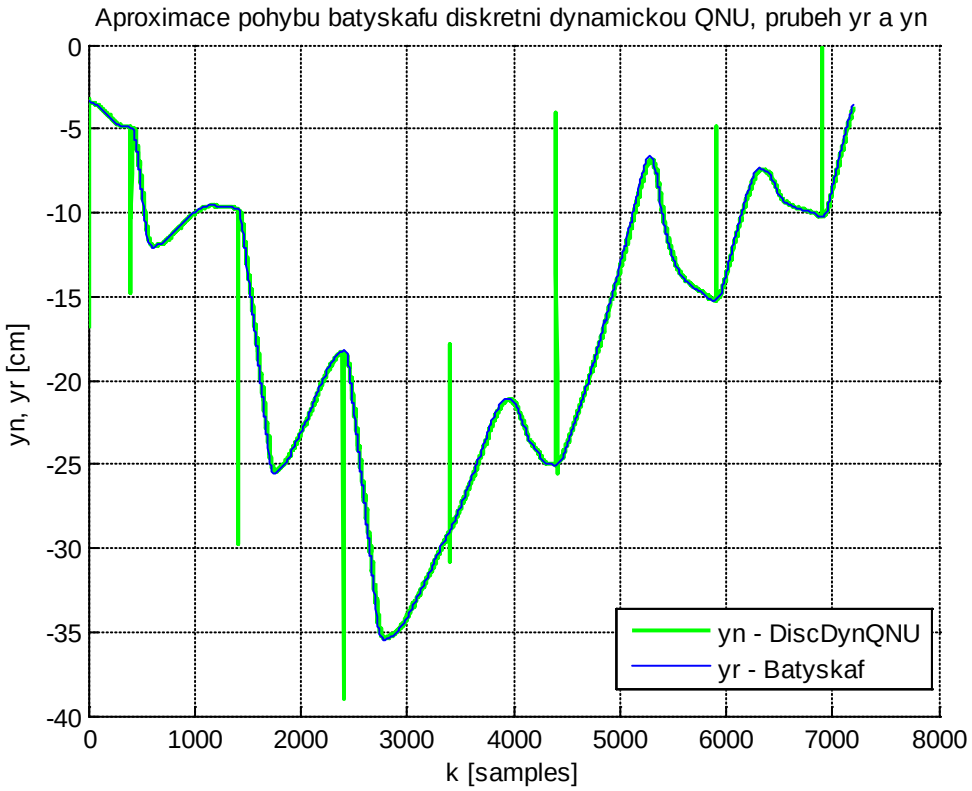
Obr. 20 - Levenberg-Marquardt algoritmus – detail neurálních vah $w_{i,j}$ (neurální váhy jednoznačně konvergují, identifikace batyskafu)

4.2.2 Adaptace - dynamická metoda

Průběžná adaptace v reálném čase, kdy se z okamžitých vstupních hodnot vypočítává hodnota výstupní, a v každém kroku se přepočítávají neurální váhy, není tak přesná jako Batch-Trainingové metody, ale zato dobře podchycuje dynamiku systému a její hlavní výhodou je právě použití v reálném čase, kdy se vstupní data mohou vlivem nějakých poruch průběžně měnit.

Zde bude ukázána identifikace soustavy batyskafu pomocí diskrétní dynamické QNU (Obr. 12), jejíž model byl zvolen v kapitole 4.1. Na rozdíl od statické QNU trénované algoritmem Levenberg-Marquardt v kapitole 4.2.1, kde vstupními daty byly přímo 3 vektory $u_{(k)}, y_{(k)}, y_{(k+1)}$, tak u dynamické QNU je vstupním vektorem pouze $u_{(k)}$ a zbylé hodnoty jsou přiváděny do agregační funkce jako zpětné vazby z výstupu QNU.

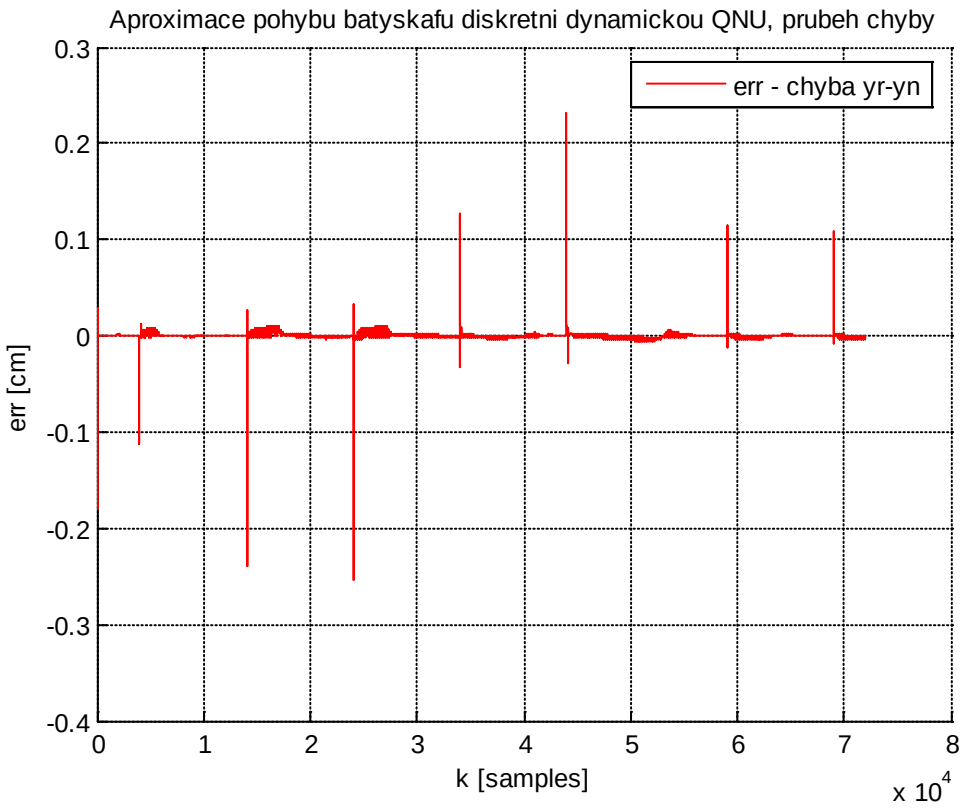
Na Obr. 21 je zobrazen průběh pohybu batyskafu y_r a aproximace pohybu batyskafu Diskrétní Dynamickou QNU (DDSOE-QNU) y_n. Je zřejmé, že DDSOE-QNU aproximuje průběh pohybu batyskafu velice dobře. Jsou zde ale viditelné špičky. Ty způsobují skokové změny vstupního signálu, které vždy rozhodí okamžité výpočty neurálních vah. Nicméně neuronová jednotka působí proti těmto změnám a snaží se je okamžitě potlačovat a minimalizovat. Tento fakt byl zjištěn experimentem, kdy byla během trvání skokových změn vstupního signálu, vypnuta ($\mu = 0$) nebo minimalizována rychlost učení. Snížení nebo vypnutí učení během špiček se projevilo déle trvajícím výkyvem nebo jeho vyšší hodnotou (ukázka porovnáním je na Obr. 25) a tedy se neprojevilo jako žádoucí. Z toho plyne, že se neuronová jednotka svým učením snaží tyto špičky minimalizovat a co nejvíce zkrátit dobu jejich trvání.



Obr. 21 – Aproximace pohybu batyskafu, průběh y_r a y_n, při identifikaci batyskafu dynamickou QNU gradientovým BP algoritmem

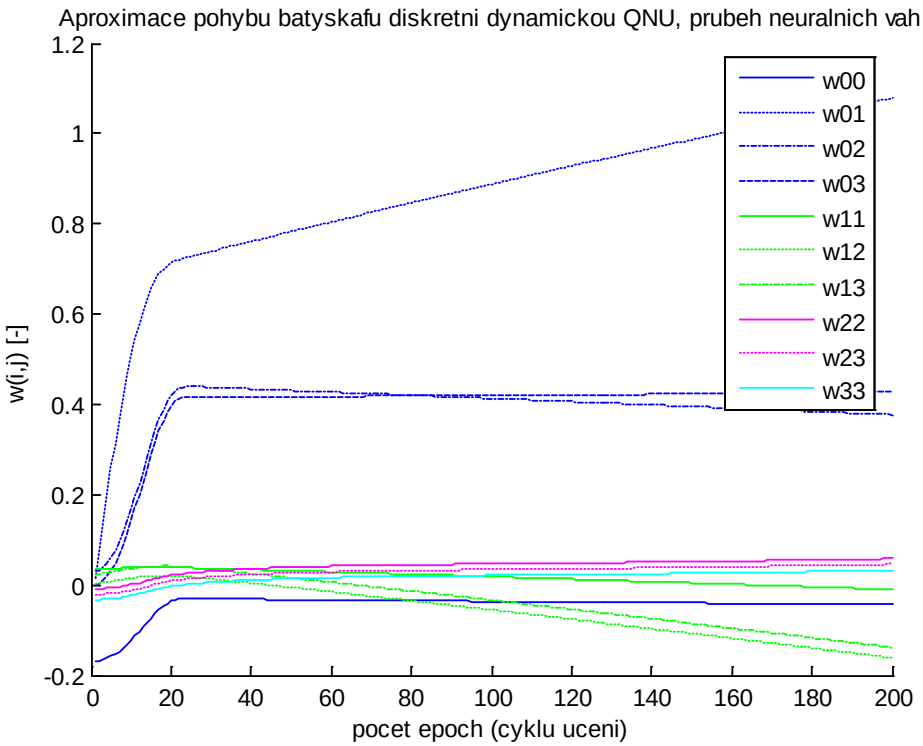
Potlačování velikosti špiček je dobře vidět na Obr. 23, kde je zobrazen průběh neurálních vah v závislosti na počtu epoch cvičení neuronové jednotky. Je vidět, že zejména neurální váhy w_{01}, w_{02} a w_{03} mají ze začátku (během prvních epoch) poměrně ostrý směr vzhůru a až přibližně po 20-ti epochách zde nastává ostrý zlom, kdy neurální váhy přestanou směřovat vzhůru a začnou se konvergovat k nějaké hodnotě. Této hodnoty můžou dosáhnout až po velice mnoha epochách, ale pro správnou identifikaci je důležité, že konvergují. A právě v tomto ostrém zlomu v průběhu neurálních vah dojde k potlačení růstu velikosti špiček, které vnáší skokový vstupní signál. Zatímco před zlomem velikosti špiček rostou, po tomto zlomu se jejich růst zastaví nebo je velice mírný. Znamená to tedy, že se DDSOE-QNU podařilo zvládnout identifikaci systému spolu se skokovými změnami vstupního tlaku.

Skokové změny (hrany) vstupních signálů jsou obecně problémem a to nejen pro neuronové jednotky, ale obecně v technické praxi. Fakt, že se neuronové jednotky svým učením snaží tyto efekty minimalizovat, svědčí v jejich prospěch.

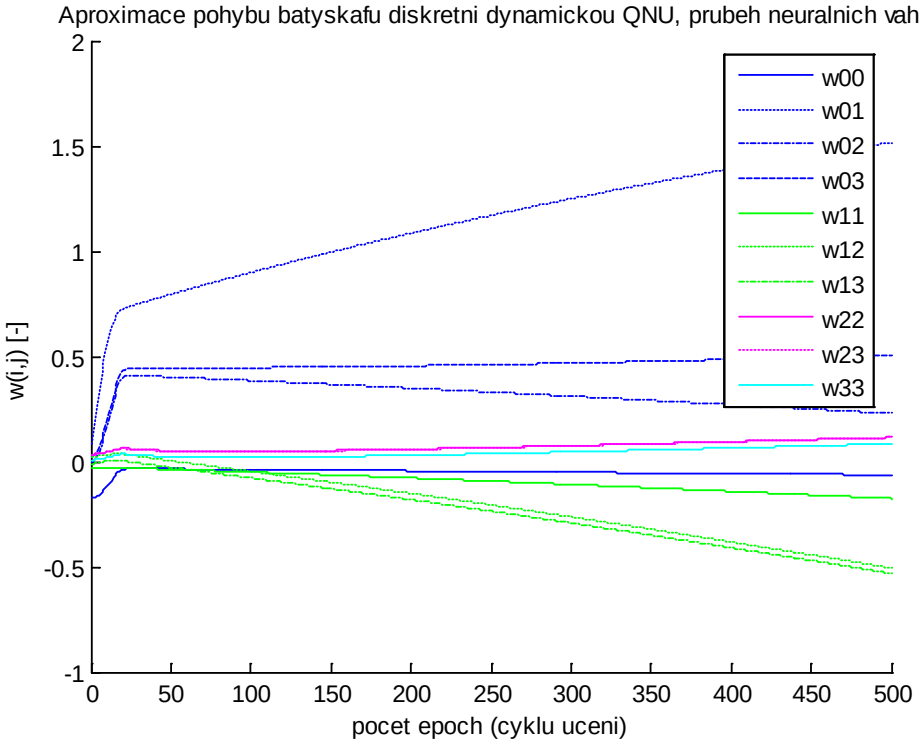


Obr. 22 – Průběh chybové veličiny při identifikaci batyskafu dynamickou QNU gradientovým BP algoritmem

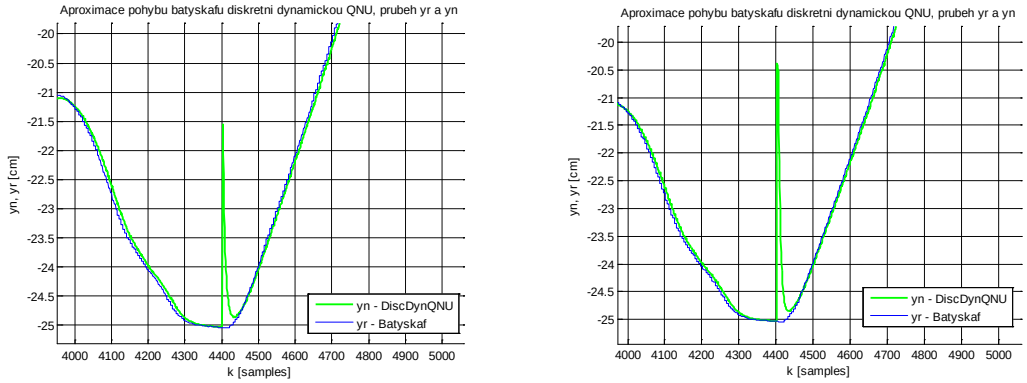
Průběh chybové veličiny je na Obr. 22 a je patrné, že chyba je velice malá (v místech, kde se neprojevují špičky od hran vstupního signálu) a dosahuje jen přibližně 6,5%, protože DDSOE-QNU velice dobře zvládá aproximaci systému.



Obr. 23 – Konvergence jednotlivých neurálních vah v závislosti na počtu epoch (200) při identifikaci batyskafu dynamickou QNU gradientovým BP algoritmem



Obr. 24 - Konvergence jednotlivých neurálních vah v závislosti na počtu epoch (500) při identifikaci batyskafu dynamickou QNU gradientovým BP algoritmem



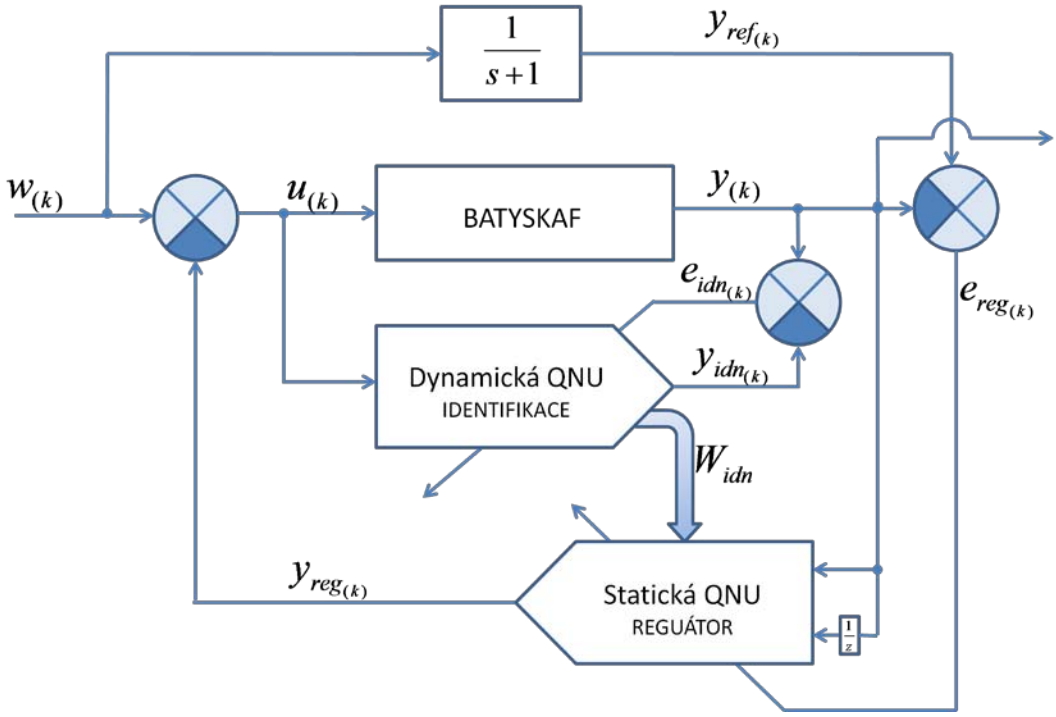
Obr. 25 – Odezva neuronové jednotky na vstupní skokovou hranu při konstantní rychlosti učení μ (vlevo) a odezva na vstupní skokovou hranu při minimalizovaném nebo nulovém μ v okamžiku skoku (vpravo), vypnutí učení při skokových změnách vstupní veličiny nezabraňuje výskytu špiček, naopak špičky jsou intenzivnější

4.2.3 Porovnání statické (LM) a dynamické (BP) tréninkové metody

Porovnání obou metod ukázalo, že i v případě, kdy se Batch-Trainingová (BT) metoda Levenberg-Marquardt (LM) velmi dobře dokázala naučit, tak se pro praktickou aplikaci ukázala jako nevhodná. Porovnáním průběhů neurálních vah (metoda LM Obr. 19, metoda BP Obr. 23) obou metod se na jednu stranu ukázalo, že průběhy neurálních vah obou metod jsou na první pohled podobné (například neurální váhy w02 a w03 společně konvergují k hodnotám 0,5 a 0,4 zatímco ostatní neurální váhy se pohybují okolo hodnoty 0), ale na druhou stranu právě mírné odlišnosti mohou hrát velice významnou roli v případě, že použijeme takto získané hodnoty jako počáteční hodnoty pro nastavení Neuro-regulátoru. Další podstatnou odlišností mezi těmito porovnávanými metodami je průběh neurální váhy w01, která se u metody LM drží spolu s ostatními okolo hodnoty 0, ale metody BP konverguje odděleně k hodnotám vyšším než 1. Právě zmíněné odlišnosti rozhodly o možnosti použití metody pro nastavení Neuro-regulátoru. Jak již bylo zmíněno, praktické experimenty ukázaly, že metoda LM je pro nastavení Neuro-regulátoru nepoužitelná, zatímco s výsledky metody BP je Neuro-regulátor funkční.

4.3 QNU jako nelineární stavový regulátor pro BATYSKAF

Jako nelineární stavový Neuro-regulátor je vhodné použít QNU (ze stejného důvodu z jakého byla provedena volba QNU pro identifikaci soustavy batyskafu). Neuro-regulátorem bude tedy diskrétní statická QNU. U klasické metody učení neuronových jednotek Back-Propagation (BP) viz Obr. 11, se provádí výpočet jednotlivých neurálních vah z hodnoty chyby rozdílu mezi výstupní hodnotou reálného systému a výstupní hodnotou z neuronové jednotky. U neuronové jednotky zapojené ve stavové zpětné vazbě se výpočet chybové hodnoty $e_{reg(k)}$ provádí z rozdílu hodnot $y_{ref(k)}$ a $y_{(k)}$ (Obr. 26), tedy z hodnot, které nejsou přímo závislé na výstupní hodnotě z této neuronové jednotky. Proto se takto získaná chybová hodnota $e_{reg(k)}$ nedá použít pro odvození výpočtu neurálních vah statické QNU.



Obr. 26 – Kompletní schéma teoretické identifikace a regulace pomocí HONNU (pro odvození)

Z tohoto důvodu je třeba využít pro odvození výpočtu neurálních vah, neurální váhy, získané při identifikaci reálného systému batyskafu diskrétní dynamickou QNU s dynamikou 2. řádu (DDSOE-QNU). Zidentifikované neurální váhy dynamické QNU

se předávají statické QNU (Neuro-regulátoru) a zde se použijí pro výpočet neurálních vah této neuronové jednotky.

Neurální váhy se mohou předávat v každém kroku, pokud by identifikace reálného systému probíhala v reálném čase. Nebo lze neurální váhy zadat do Neuro-regulátoru napevno, pokud bychom provedli identifikaci odděleně od běžícího procesu.

Odvození výpočtu neurálních vah Neuro-regulátoru

$$\text{Definice:} \qquad D\{y_{(k)}\} \cong y_{(k+1)} \tag{28}$$

Odvození přírůstků jednotlivých neurálních vah diskrétní statické QNU bude provedeno podle Obr. 26. Výstupní hodnota z diskrétní dynamické QNU $y_{idn(k)}$ se vypočte podle (29) a výstupní hodnota $y_{reg(k)}$ podle (30).

$$y_{(k)} \approx y_{idn\,(k)} = DD\{f_{idn}\,(u_{(k)}, y_{(k)}, y_{(k+1)}, W_{idn}\,)\} \tag{29}$$

$$y_{reg\,(k)} = f_{reg}\,(\,y_{(k)}, y_{(k-1)}, W_{reg}\,) \tag{30}$$

Akční veličina $u_{(k)}$ je rozdílem požadované veličiny $w_{(k)}$ a veličiny $y_{reg(k)}$ (31). Veličinu $u_{(k)}$ pak dosazením zpět do (29) získáme vztah pro výslednou výstupní hodnotu (32).

$$u_{(k)} = w_{(k)} - y_{reg\,(k)} \tag{31}$$

$$y_{(k)} = DD\left\{f_{idn}\,\left(w_{(k)} - y_{reg\,(k)}, y_{(k)}, y_{(k+1)}, W_{idn}\,\right)\right\} \tag{32}$$

Podle metody Back-Propagation (BP), popsané v kapitole 3.3.3 se podle vztahů (33) a (34) vypočte změna neurálních vah v každém kroku. Aplikací těchto vztahů na rovnici (32) získáme pro výpočty změn neurálních vah vztah (35).

$$\Delta w_{ij} = -\mu \frac{\delta J}{\delta w_{ij}} \tag{33}$$

$$J = \frac{1}{2}\mathrm{e}^2 \tag{34}$$

$$\Delta w_{i,j} = \mu \cdot e \cdot DD\left\{\frac{\partial y_{(k)}}{\partial w_{i,j}}\right\} \tag{35}$$

Po detailnějším rozepsání vztahu (35) získáme výsledný obecný vztah (36) pro přírůstky neurálních vah Neuro-regulátoru.

$$\Delta w_{i,j} = \mu \cdot e \cdot DD\left\{\frac{\partial y_{(k)}}{\partial y_{reg\,(k)}} \cdot \frac{\partial y_{reg\,(k)}}{\partial w_{i,j}}\right\} = \mu \cdot e \cdot DD\left\{\frac{\partial y_{(k)}}{\partial u_{(k)}} \cdot \frac{\partial u_{(k)}}{\partial y_{reg\,(k)}} \cdot \frac{\partial y_{reg\,(k)}}{\partial w_{i,j}}\right\} \tag{36}$$

Dále zde bude provedeno konkrétní odvození přírůstků neurálních vah aplikací obecného výsledného vztahu (36) na agregační funkci diskrétní dynamické QNU pro identifikaci soustavy batyskafu a na agregační funkci diskrétní statické QNU jako Neuro-regulátoru soustavy. Obecný vztah pro QNU je uveden v (38) a vektory vstupů jsou uvedeny v (37).

$$\begin{aligned} \boldsymbol{x_a^{idn}} &= \left[x_0\ u_{(k)}\ y_{(k)}\ y_{(k+1)}\right]^T \\ \boldsymbol{x_a^{reg}} &= \left[x_0\ y_{(k)}\ y_{(k-1)}\right]^T \end{aligned} \tag{37}$$

$$f_{QNU} = \sum_{i=0}^{n-1}\sum_{j=i}^{n-1} x_i x_j w_{ij} \tag{38}$$

, kde $n = 4$ *(počet vstupních hodnot) a* $u_0 = 1$ *(prahová hodnota)*

Rozepsané agregační funkce pro jednotlivé neuronové jednotky jsou uvedeny v (39) a (40).

$$\begin{aligned} y_{(k)} \approx f_{idn\,(k)} = {} & w_{00} + w_{01}u_{(k)} + w_{02}y_{(k)} + w_{03}y_{(k+1)} + w_{11}u_{(k)}^2 \\ & + w_{12}u_{(k)}y_{(k)} + w_{13}u_{(k)}y_{(k+1)} + w_{22}y_{(k)}^2 + w_{23}y_{(k)}y_{(k+1)} \\ & + w_{33}y_{(k+1)}^2 \end{aligned} \tag{39}$$

$$\begin{aligned} f_{reg\,(k)} = {} & w_{00} + w_{01}y_{(k)} + w_{02}y_{(k-1)} + w_{11}y_{(k)}^2 + w_{12}y_{(k)}y_{(k-1)} \\ & + w_{22}y_{(k-1)}^2 \end{aligned} \tag{40}$$

Do rovnice agregační funkce diskrétního dynamického QNU pro identifikaci dosadíme za všechna $u_{(k)}$ podle vztahu (31) a aplikujeme část parciálních derivací $\frac{\partial y_{(k)}}{\partial u_{(k)}} \cdot \frac{\partial u_{(k)}}{\partial y_{reg\,(k)}}$ ze vztahu (36). Po tomto kroku získáme část vztahu, ve kterém se budou vyskytovat neurální váhy diskrétní dynamické QNU spolu s jejími vstupními hodnotami (41). Tato část vztahu bude pro další výpočet neurálních vah diskrétní statické QNU (Neuro-regulátoru) shodná pro všechny její neurální váhy. Hodnota $y_{(k+1)}$ je počítána diskrétní dynamickou QNU.

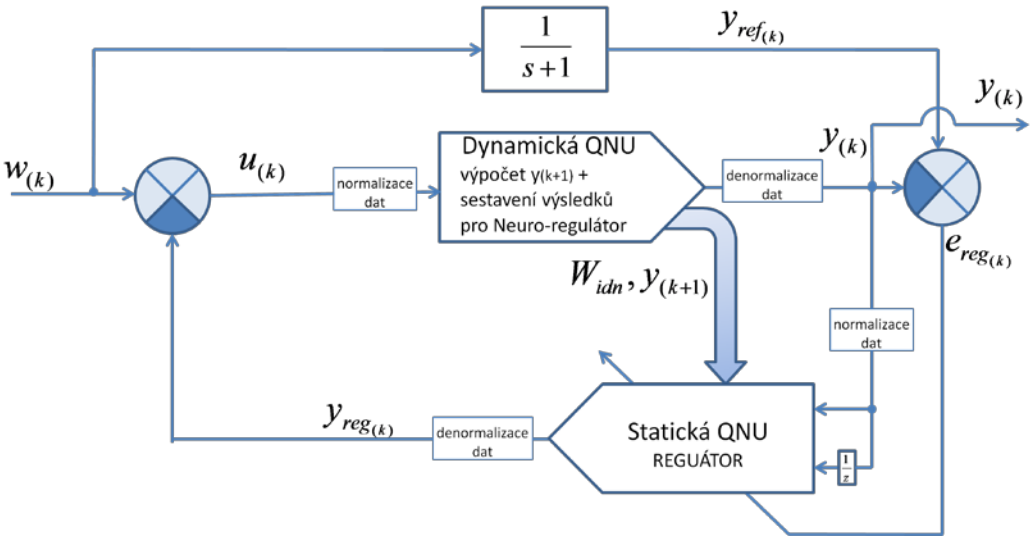
$$\left[-w_{01} - 2w_{11}u_{(k)} - w_{12}y_{(k)} - w_{13}y_{(k+1)}\right] \tag{41}$$

Nyní bude provedena aplikace druhé části parciálních derivací $\frac{\partial y_{reg\,(k)}}{\partial w_{i,j}}$ ze vztahu (36) na agregační funkci diskrétní statické QNU. Získanými členy u jednotlivých neurálních vah vynásobíme členy uvedené ve vztahu (41) a celé dosadíme zpět do vztahu (36). Výsledné změny neurálních vah pro diskrétní statickou QNU (Neuro-regulátor) jsou uvedeny v (42).

$$\begin{aligned}
\Delta w_{00}^{reg} &= \mu \cdot e \cdot DD\left\{\left[-w_{01} - 2w_{11}u_{(k)} - w_{12}y_{(k)} - w_{13}y_{(k+1)}\right] \cdot 1\right\} \\
\Delta w_{01}^{reg} &= \mu \cdot e \cdot DD\left\{\left[-w_{01} - 2w_{11}u_{(k)} - w_{12}y_{(k)} - w_{13}y_{(k+1)}\right] \cdot y_{(k)}\right\} \\
\Delta w_{02}^{reg} &= \mu \cdot e \cdot DD\left\{\left[-w_{01} - 2w_{11}u_{(k)} - w_{12}y_{(k)} - w_{13}y_{(k+1)}\right] \cdot y_{(k-1)}\right\} \\
\Delta w_{11}^{reg} &= \mu \cdot e \cdot DD\left\{\left[-w_{01} - 2w_{11}u_{(k)} - w_{12}y_{(k)} - w_{13}y_{(k+1)}\right] \cdot y_{(k)}^2\right\} \\
\Delta w_{12}^{reg} &= \mu \cdot e \cdot DD\left\{\left[-w_{01} - 2w_{11}u_{(k)} - w_{12}y_{(k)} - w_{13}y_{(k+1)}\right] \cdot y_{(k)}y_{(k-1)}\right\} \\
\Delta w_{22}^{reg} &= \mu \cdot e \cdot DD\left\{\left[-w_{01} - 2w_{11}u_{(k)} - w_{12}y_{(k)} - w_{13}y_{(k+1)}\right] \cdot y_{(k-1)}^2\right\}
\end{aligned} \tag{42}$$

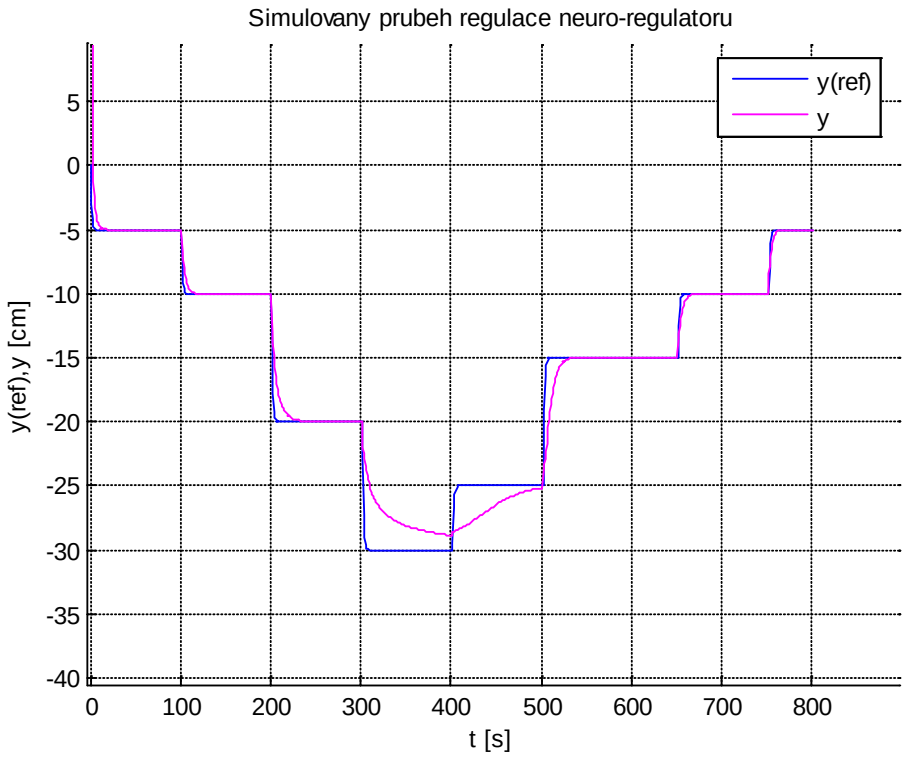
4.3.1 Simulace regulace soustavy Neuro-regulátorem (teoretická)

Nyní je možné simulačně otestovat odvozený Neuro-regulátor. Diskrétní dynamickou QNU, která byla použita pro identifikaci systému batyskafu, nahradíme celý systém batyskafu. Protože je tato neuronová jednotka již vycvičena, tak její systém by měl aproximovat kompletní dynamický systém batyskafu. Jednotka se již nebude dál učit a její neuronové váhy budou nastaveny napevno a to tak, že použijeme poslední hodnoty neurálních vah, které jsme získali při cvičení neuronové jednotky (například po 1500 epochách). Do stavové zpětné vazby zapojíme diskrétní statickou QNU jako stavový zpětnovazební regulátor. Schéma simulačního zapojení je na Obr. 27.

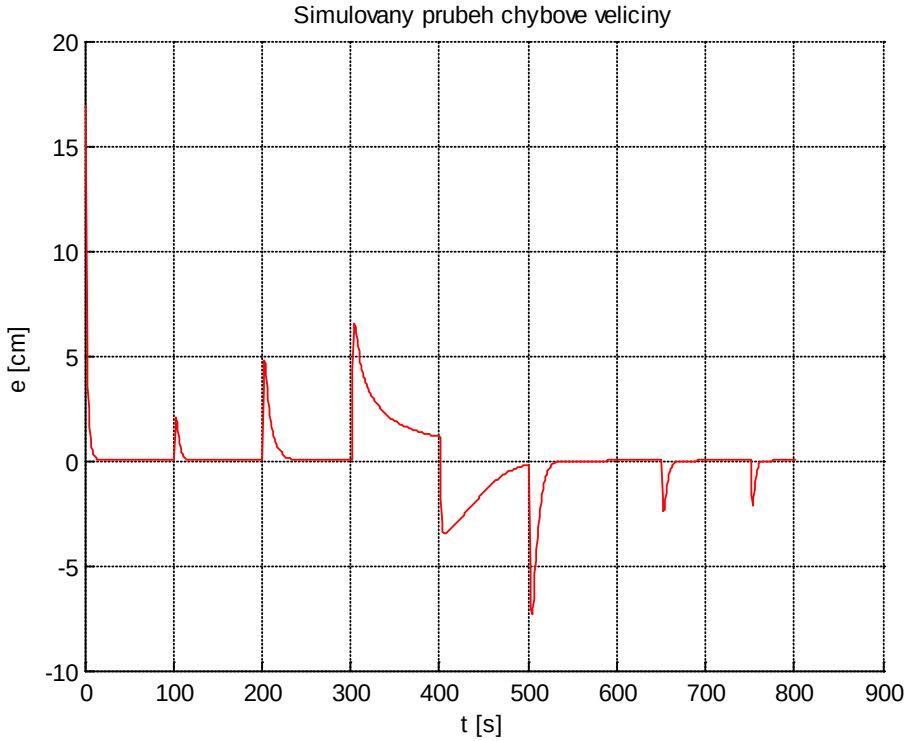


Obr. 27 – Simulační schéma Neuro-regulátoru

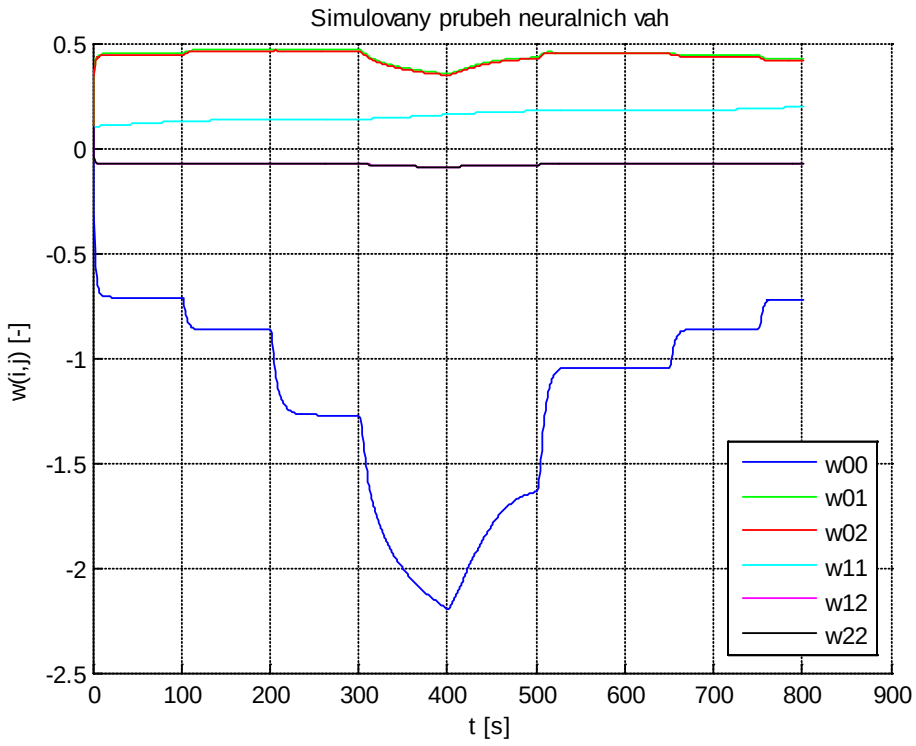
Neurální váhy W_{idn} se předávají podle odvozených vztahů. Chyba $e_{reg(k)}$ se vypočítává z rozdílu mezi $y_{ref(k)}$ a $y_{(k)}$, kde $y_{ref(k)}$ stanovuje model přenosu reálné soustavy, podle kterého chceme, aby se regulátor choval. Neuro-regulátor by se pak měl snažit regulovat soustavu tak, aby přechodová charakteristika regulovaného obvodu měl shodný průběh jako zadaný přenos $\frac{1}{s+1}$. Výsledné důležité veličiny a jejich průběhy zachycují následující obrázky. Na Obr. 28 je průběh regulace pohybu batyskafu $y_{(k)}$, který se snaží kopírovat průběh referenční hodnoty $y_{ref(k)}$. Dá se říci, že se to vcelku dobře daří, až na hodnoty mezi -25 a -30 cm hloubky, kde by bylo třeba prodloužit skok, aby hodnota $y_{(k)}$ mohla dosáhnout žádané hodnoty. Jedná se o pohyb batyskafu v takřka maximální hloubce a tyto odchylky mohou být způsobené nepřesnostmi soustavy nebo neurálních vah při identifikaci i regulaci. Ale i tak lze výsledný simulovaný (teoretický) průběh považovat za velice uspokojivý. Na Obr. 29 je zobrazen průběh chybové veličiny $e_{reg(k)}$ a na Obr. 30 jsou průběhy neurálních vah diskrétní statické QNU.



Obr. 28 – Simulace regulace Neuro-regulátoru, průběh $y_{ref(k)}$ a $y_{(k)}$



Obr. 29 – Průběhu chybové veličiny $e_{reg(k)}$ při simulaci regulace Neuro-regulátoru

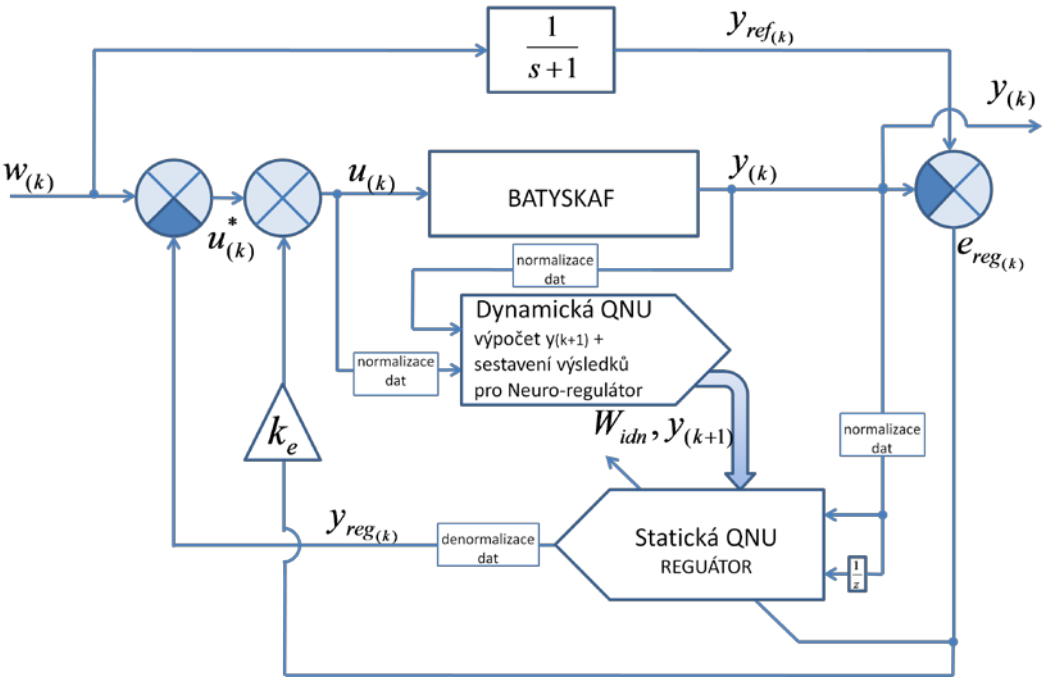


Obr. 30 – Průběh neurálních vah při simulaci regulace diskrétní statické QNU jako Neuro-regulátoru

4.3.2 Reálná regulace soustavy Neuro-regulátorem (praktická)

Nyní bude ukázána reálná regulace soustavy batyskafu, která odvozením vychází z teoretické části a byla pro praktické účely mírně modifikována. Reálná regulace soustavy batyskafu bude realizována podle schématu, které je uvedeno na Obr. 31. Popíšeme důležité části schématu Neuro-regulátoru. K soustavě batyskafu je paralelně připojena diskrétní dynamická QNU s dynamikou 2. řádu, která zajišťuje průběžné výpočty hodnot, které pak spolu neurálními váhami předává v každém kroku dál diskrétní statické QNU. Neurální váhy neuronové jednotky pro identifikaci byly nastaveny napevno a jsou výsledkem učení adaptací při identifikaci soustavy batyskafu během 1500 epoch. Do statické diskrétní QNU, která plní funkci samotného Neuro-regulátoru, vstupuje měřená reálná hodnota ze soustavy batyskafu $y_{(k)}$ a $y_{(k+1)}$, dále pak vstupují vypočtené neurální váhy z diskrétní dynamické QNU W_{idn} a chyba $e_{reg(k)}$, která je rozdílem mezi referenční hodnotou $y_{ref(k)}$ a výstupní hodnotou batyskafu $y_{(k)}$. Výstup

z Neuro-regulátoru je jako stavová zpětná vazba porovnán s žádanou veličinou $w_{(k)}$ a výsledkem je akční veličina $u^*_{(k)}$.



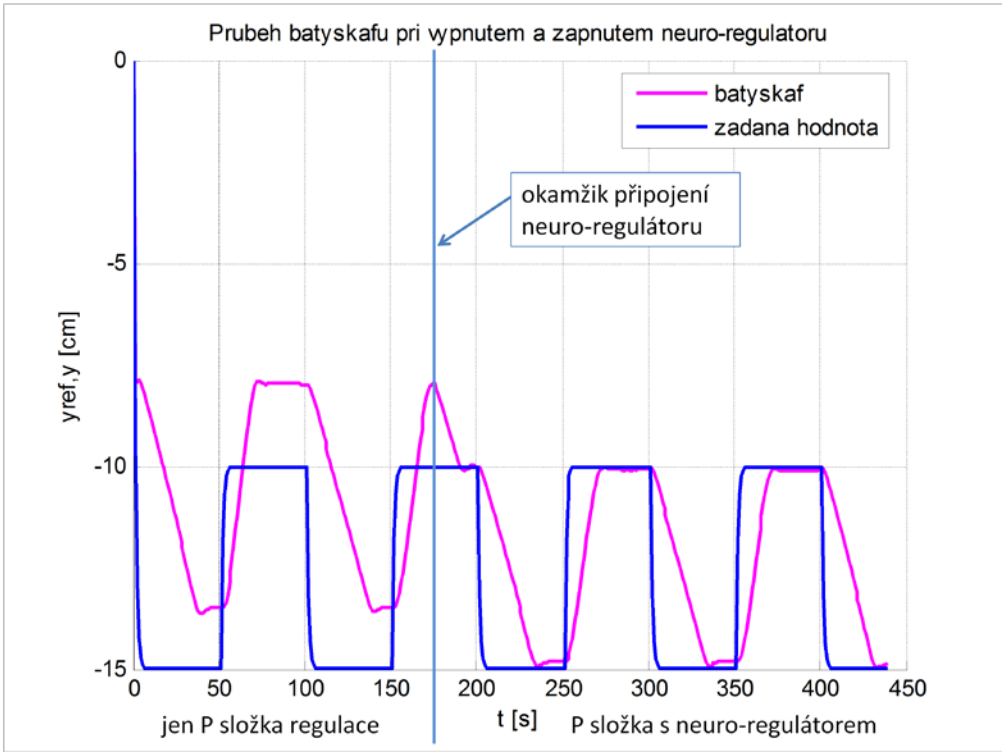
Obr. 31 – Reálné a funkční schéma Neuro-regulátoru batyskafu

Z důvodu posílení rychlosti změn akční veličiny byla ještě provedena modifikace pomocí chybové veličiny $e_{reg(k)}$, která byla přivedena na vstup do batyskafu tak, že se její hodnota přičítá k akční veličině $u^*_{(k)}$ a výsledkem je akční veličina $u_{(k)}$. Protože chybová veličina $e_{reg(k)}$ nabývá hodnot kladných i záporných (podle toho jestli je $y_{(k)}$ větší nebo menší než $y_{ref(k)}$), stává se z této zpětné vazby proporcionální složka. Nastavení její velikosti lze provést změnou konstanty k_e. Změnou hodnoty konstanty k_e lze mírně doladit odezvu soustavy při regulaci - má na jedné straně mírný vliv na přesnost regulace, ale na druhé straně může způsobit velké překmity nebo stálé drobné kmitání.

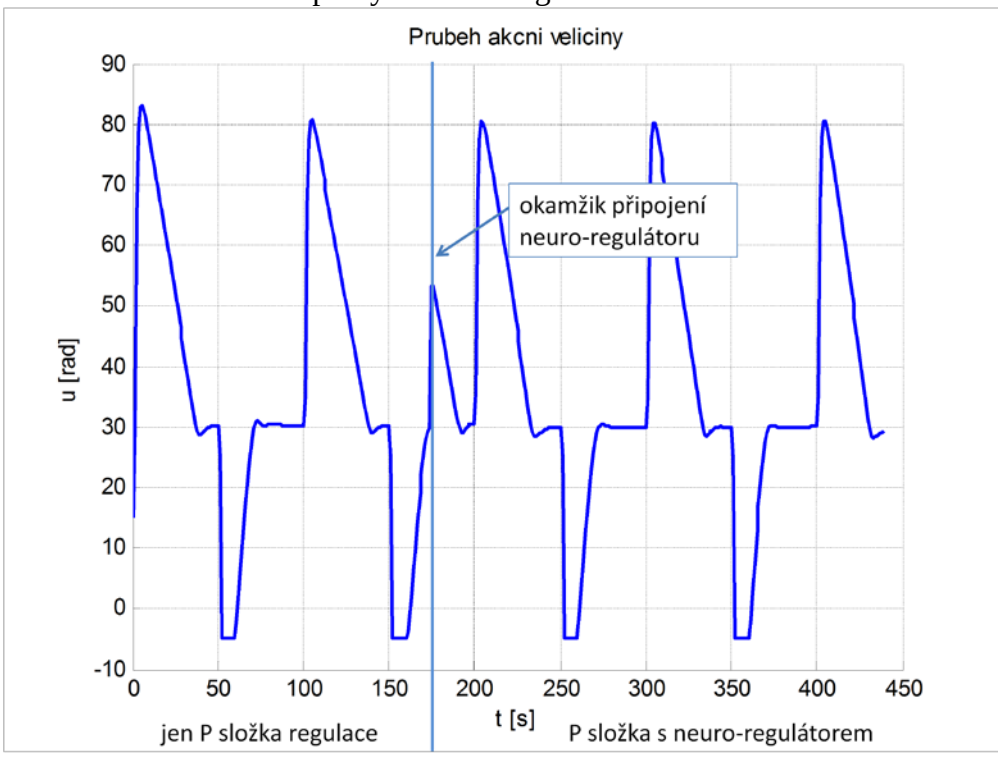
Samotná regulace s použitím jen proporciální složky P odvíjející se od chybové veličiny $e_{reg(k)}$ a při odpojeném Neuro-regulátoru probíhá sice bez překmitů, ale nereaguje na nelinearity soustavy batyskafu, proto se takto regulovaná výstupní veličina $y_{(k)}$ neshoduje po ustálení hodnotami s žádanou hodnotou veličiny $w_{(k)}$. Navíc pro různé hloubky regulace a pro různé změny žádané veličiny se tato odchylka mění.

Na Obr. 32 je porovnání odezvy soustavy batyskafu při regulaci od proporciální složky P s vypnutým Neuro-regulátorem (v první části grafu) a dále se zapnutým Neuro-regulátorem (druhá část grafu). Porovnání bylo provedeno na vstupní skokové

změny žádané veličiny mezi hloubkou 10 a 15 cm. Je vidět, že samotná proporciální složka P reguluje soustavu špatně – odchylka od žádané veličiny $w_{(k)}$ a regulované veličiny $y_{(k)}$ je velice velká a dosahuje až hodnoty 2 cm.

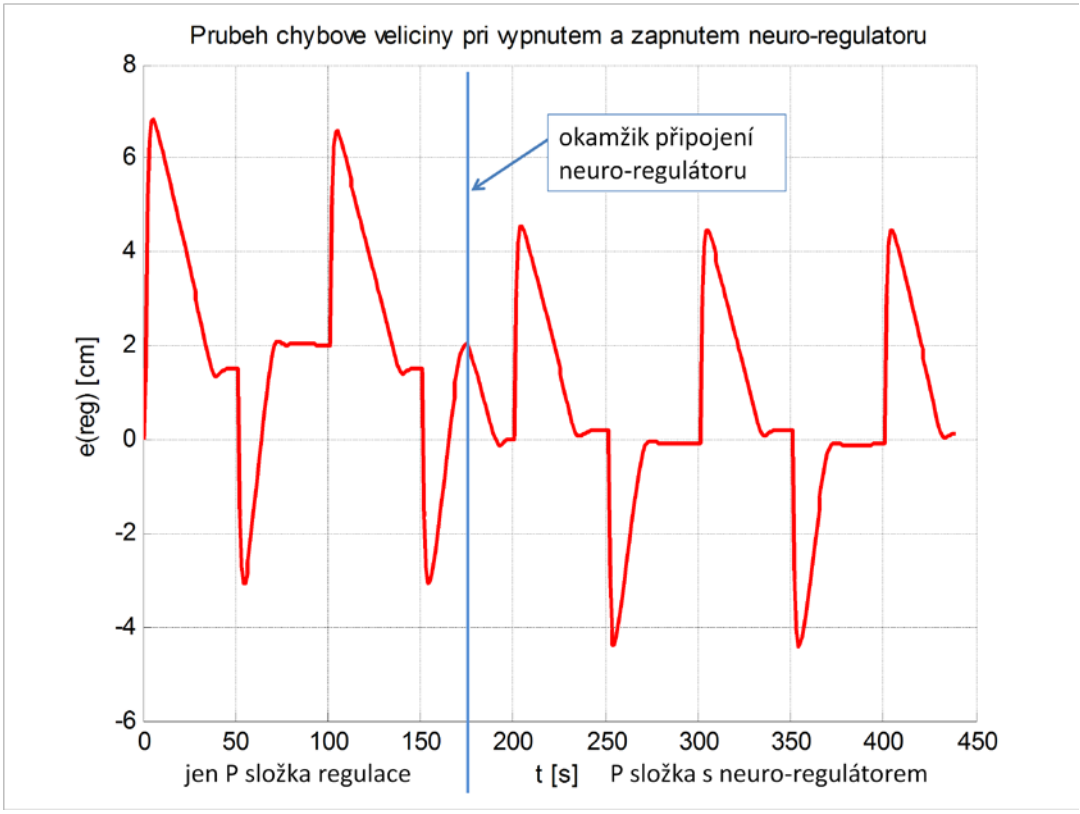


Obr. 32 – Odezva batyskafu na skokové změny žádané veličiny s vypnutým a následně zapnutým Neuro-regulátorem



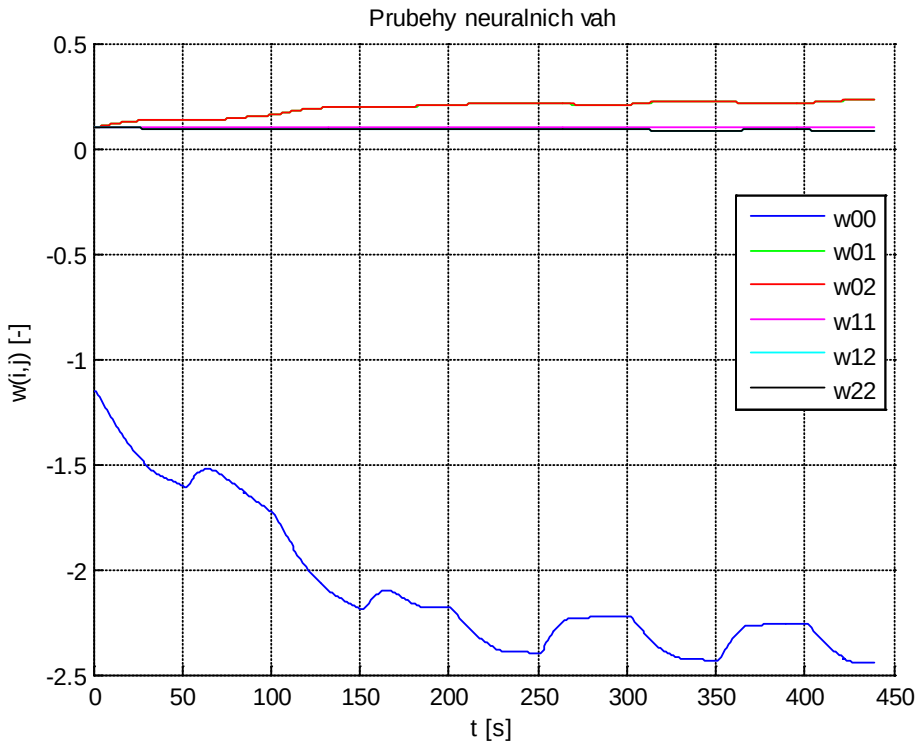
Obr. 33 – Průběh akční veličiny *u* při odpojeném a připojeném Neuro-regulátoru

V okamžiku připojení Neuro-regulátoru, který dokáže aproximovat nelinearity systému, se průběhy žádané a regulované veličiny srovnají. Stejně kvalitní regulaci zvládne Neuro-regulátor v jakékoli hloubce, z jakékoli počáteční polohy a pro jakkoli velkou změnu žádané veličiny. Průběh akční veličiny $u_{(k)}$ je na Obr. 33 a chybové veličiny $e_{reg(k)}$ na Obr. 34.



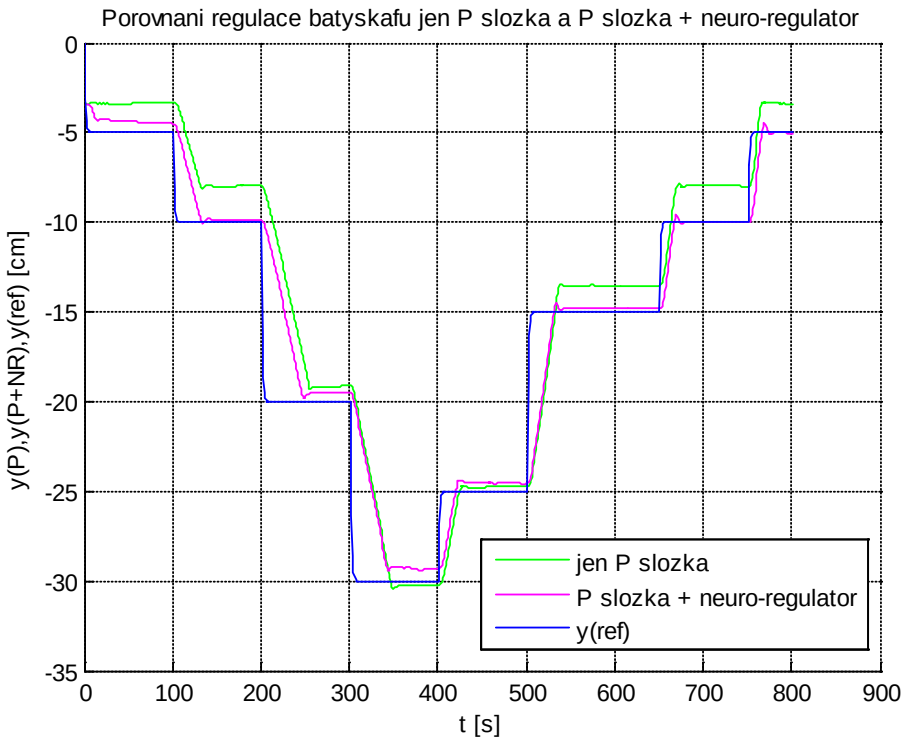
Obr. 34 – Průběh chybové veličiny $e_{reg(k)}$ při odpojeném a připojeném Neuro-regulátoru

Neuro-regulátor se v průběhu regulace reálného systému stále adaptuje. Stále dolaďuje svoje neurální váhy a zdokonaluje tak svou regulaci a aproximaci systému. Na Obr. 35 jsou neurální váhy Neuro-regulátoru (reagující na regulaci skokových změn podle Obr. 32), které s průběhem času mění a je vidět, že konvergují k nějakým reálným hodnotám. To je důkazem toho, že se Neuro-regulátor stále adaptuje.

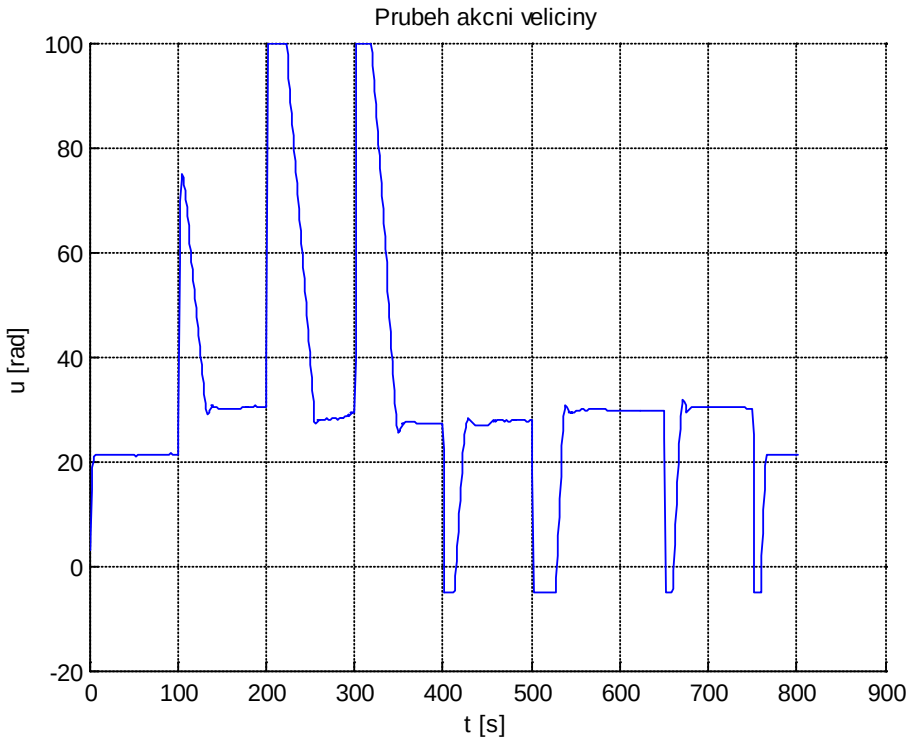


Obr. 35 – Průběhy neurálních vah Neuro-regulátoru při regulaci skokových změn (pozn.: některé neurální váhy konvergují takřka stejnoměrně, překrývají se a nejsou vidět)

Porovnání regulace samotné proporciální složky P a Neuro-regulátoru s P složkou je, pro různé skokové změny žádané hloubky směrem dolů i vzhůru, uvedeno na Obr. 36. Opět je vidět, že samotná proporcionální složka nezvládne regulovat tak přesně jako spolu s Neuro-regulátorem, který reaguje na dynamiku soustavy. Na Obr. 37 je uveden průběh akční veličiny se zapojeným Neuro-regulátorem. Zde je třeba zmínit zapojení saturačního členu mezi akční veličinu a vstup do soustavy batyskafu a do diskrétní dynamické QNU (viz Obr. 44 v příloze). Saturační člen má zde spíš pomocnou úlohu a má zabránit případné možnosti rozkmitání a destabilizace soustavy regulace. Serva soustavy reagují jen na akční zásah v rozsahu hodnot 0 až 40 radiánů, které reprezentují natočení serva a tím i otevření ventilu tlaku. Proto saturační člen omezuje zbytečně velké nebo nízké akční zásahy (na Obr. 37 jsou proto oříznuté horní a dolní špičky).

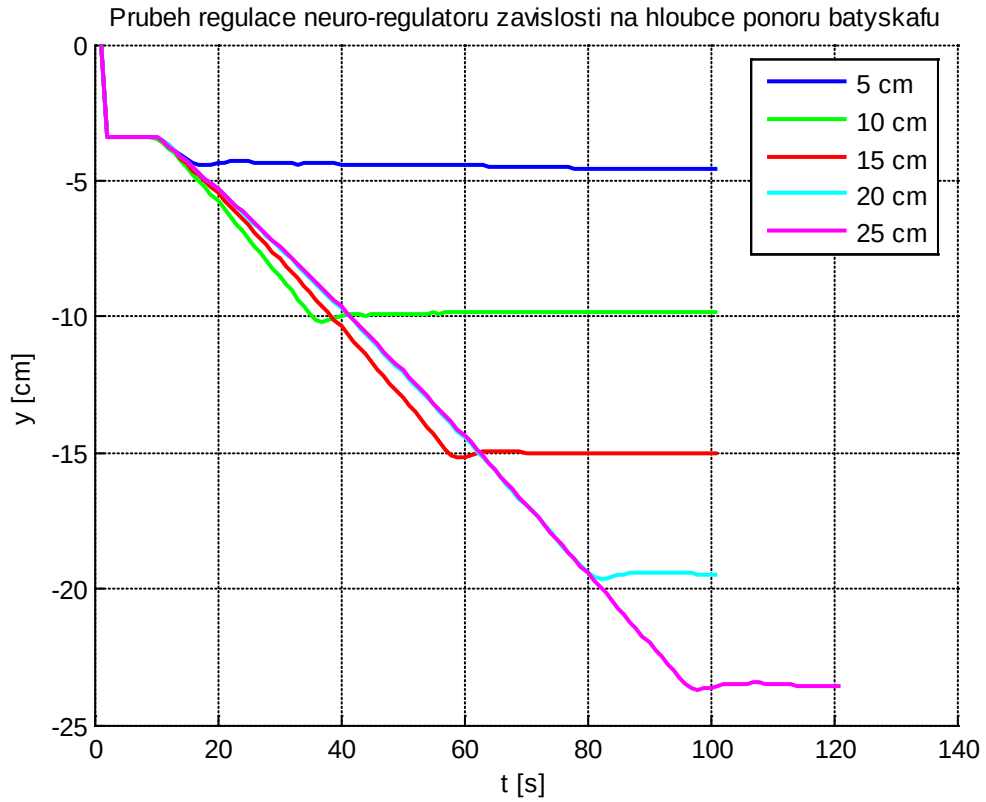


Obr. 36 – Porovnání regulace Neuro-regulátorem a samotné proporciální P složky ze stavové zpětné vazby



Obr. 37 – Průběh akční veličiny *u* při regulaci Neuro-regulátoru

Dále budou ukázány průběhy odezvy systému batyskafu při regulaci Neuro-regulátorem na požadované hodnoty hloubky postupně 5, 10, 15, 20 a 25 cm. Podle očekávání dosáhl batyskaf požadovaných hloubek v co nejkratším čase a bez překmitu. Odezvy soustavy batyskafu jsou zobrazeny na Obr. 38. Nepřesné ustálení při ponoru na hodnoty 20 a 25 může být dáno nepřesností v identifikaci soustavy batyskafu, ale tyto nepřesnosti mohou být snadno odstraněny buď jednorázově (změnou nastavení parametrů Neuro-regulátoru nebo jeho proporciální složky) nebo zpřesněním identifikace soustavy.

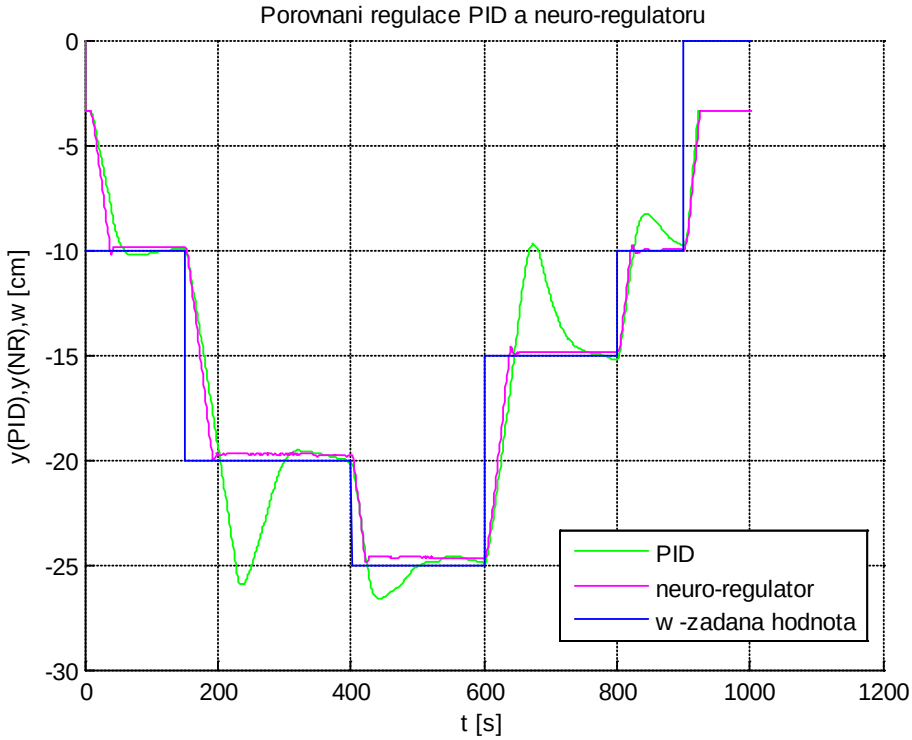


Obr. 38 - Průběhy regulace Neuro-regulátoru v závislosti na hloubce ponoru batyskafu

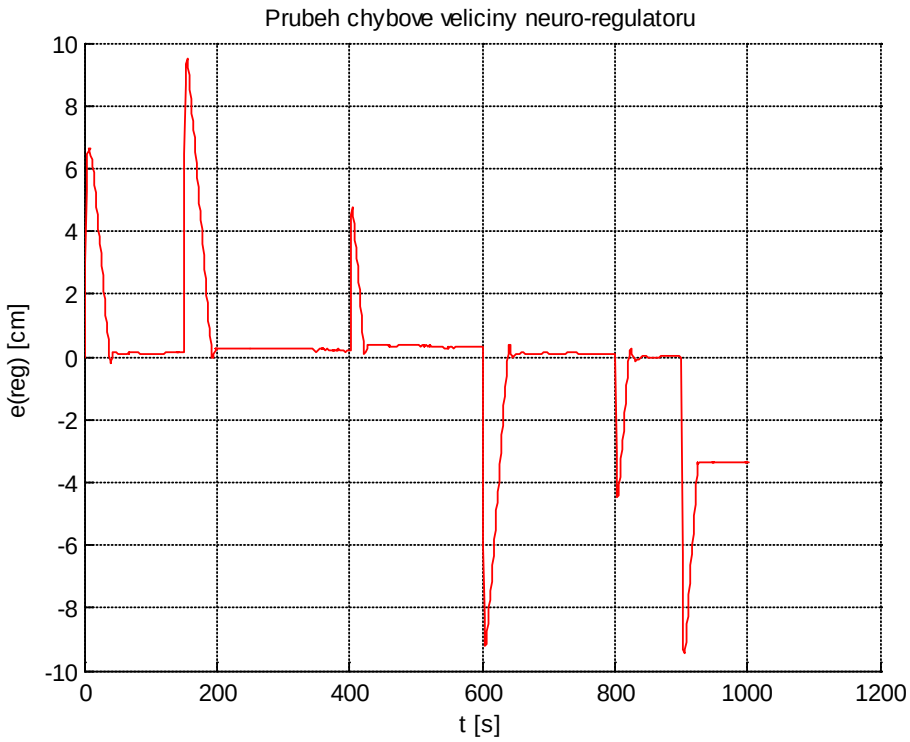
5 Výsledky

Na Obr. 39 je porovnání odezvy systému batyskafu při regulaci klasickým PID regulátorem a při regulaci adaptabilním Neuro-regulátorem. Změny žádané veličiny jsou skokové a nepravidelné, proto se zde nejvíc projeví výborná regulace Neuro-regulátoru, který reguluje takřka stejně dobře v celém rozsahu. Průběh chybové veličiny $e_{reg(k)}$ je na Obr. 40. Za povšimnutí stojí fakt, že u průběhu chyby jde jen o krátké špičky, v okamžiku změny žádané veličiny, které se rychle srovnají zpět na nulovou hodnotu. Z toho vyplývá, že regulace je rychlá a po jakékoli změně žádané veličiny se batyskaf rychle dostane na požadovanou hloubku.

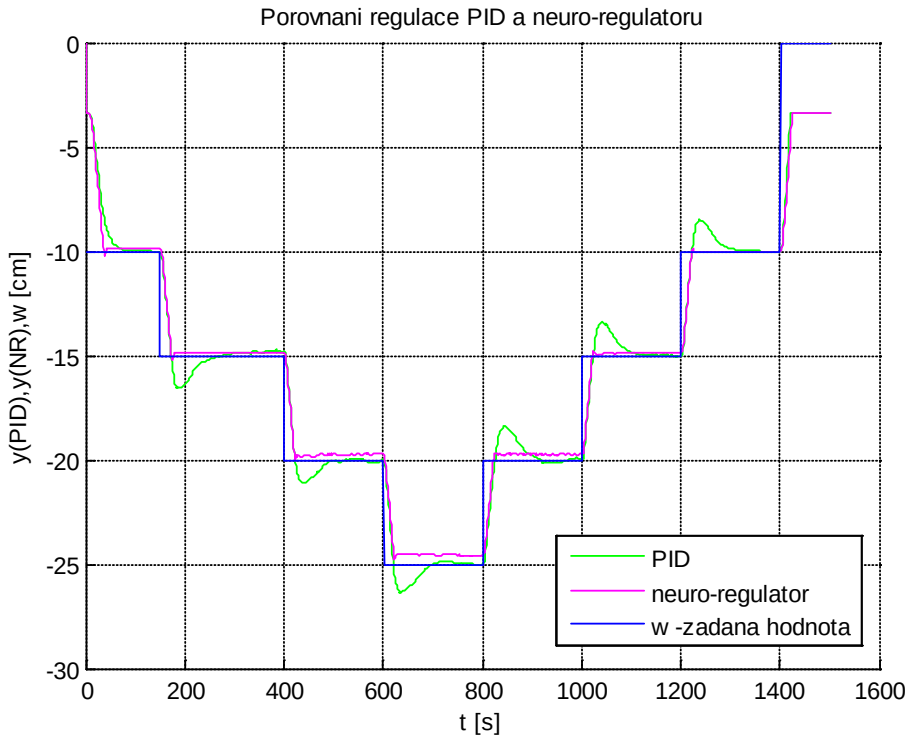
Odezva soustavy batyskafu při regulaci PID regulátorem a Neuro-regulátorem pro krátké a stejně velké změny žádané veličiny je uvedena na Obr. 41. Zde je dobře vidět, že pro krátké a stejně dlouhé změny žádané veličiny se soustava batyskafu chová s PID regulátorem stejně – překmity PID regulátoru i jejich velikosti jsou podobné. To potvrzuje i další experimenty, že u soustavy batyskafu záleží víc na velikosti změn žádané veličiny a než na hloubce ponoru. Nicméně, jak již bylo zmíněno, hloubka a počáteční poloha má také vliv na nelineárnost soustavy.



Obr. 39 – Porovnání regulace PID a Neuro-regulátoru, větší skokové změny žádané veličiny (PID regulátor vyladěn na hodnotu hloubky 10 cm)



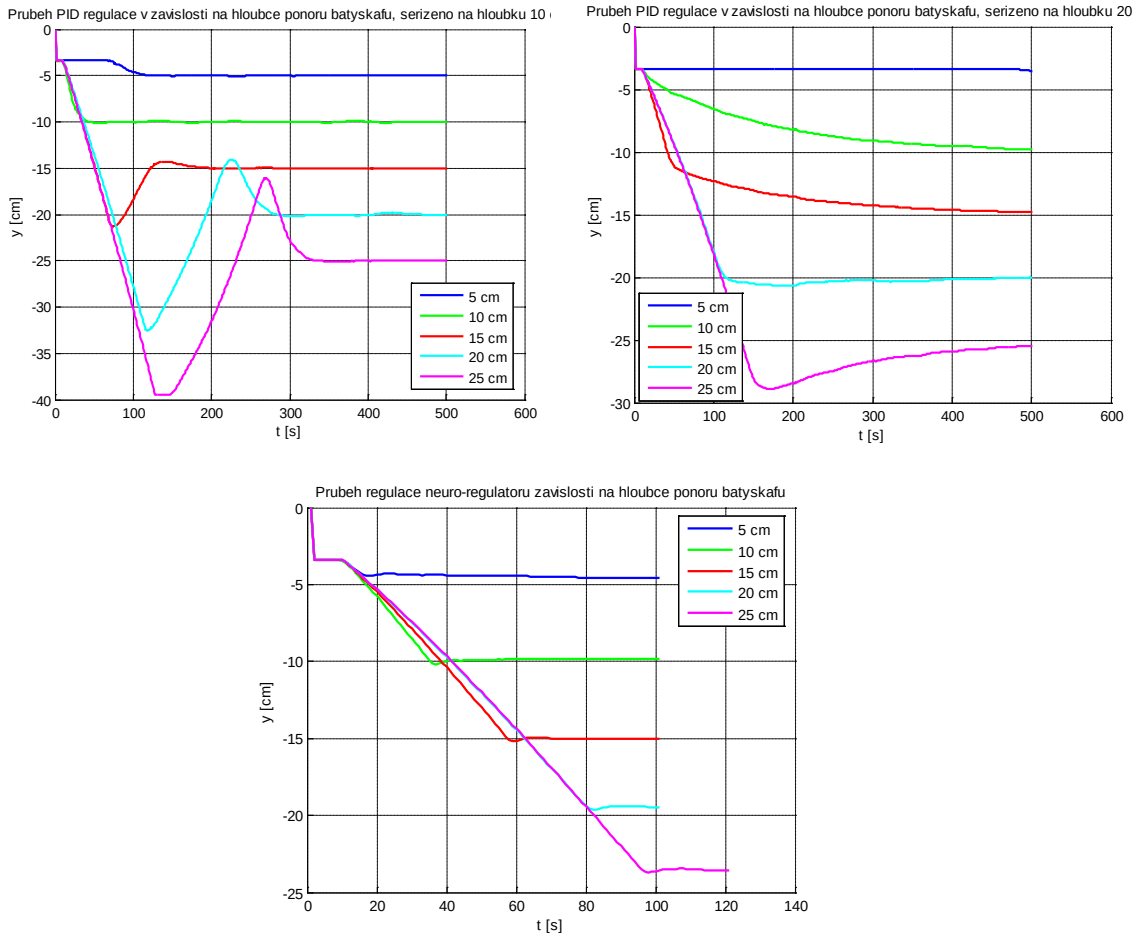
Obr. 40 – Průběh chybové veličiny Neuro-regulátoru, větší skokové změny žádané veličiny



Obr. 41 - Porovnání regulace PID a Neuro-regulátoru, menší skokové změny žádané veličiny

Všechny uvedené důvody a experimenty ukazují, že regulace klasickým PID regulátorem je pro tuto nelineární soustavu nedostatečná. Následující kompletní porovnání odezvy soustavy batyskafu pro oba regulátory opět potvrzuje použitelnost Neuro-regulátoru.

Na Obr. 42 jsou uvedeny odezvy soustavy batyskafu ve třech obrázcích a vždy jde o regulaci pohybu batyskafu z maximální horní výchozí polohy do žádané hloubky, která se postupně mění. Nejdříve je PID regulátor vyladěn na regulaci v hloubce 10 cm, na dalším obrázku je vyladěn pro hloubku 20 cm a na třetím obrázku je regulace s Neuro-regulátorem, který zvládá regulovat v celém rozsahu.



Obr. 42 – Porovnání regulace klasického PID regulátoru a Neuro-regulátoru, PID seřízený pro hloubku 10 cm (vlevo nahoře), PID seřízený pro hloubku 20 cm (vpravo nahoře) a regulace Neuro-regulátorem (dole)

Jednotlivé odezvy z obrázků již byly popsány jednotlivě v předchozích kapitolách, zde se jedná pouze o rekapitulaci a porovnání průběhů najednou. Je patrné, že PID regulátor reguluje dobře jen v malém okolí svého pracovního bodu a pro změnu

hloubky by musel být přeladěn. Naopak Neuro-regulátor se přelaďovat nemusí a reguluje v celém rozsahu.

6 Závěr

Cílem této práce bylo navrhnout adaptabilní Neuro-regulátor, který by byl aplikovatelný na úlohy automatického řízení. Zejména pro řízení laboratorní úlohy Batyskaf, která je ve své podstatě nelineární a klasické PID regulátory nezvládnou dobrou regulaci v celém rozsahu regulované veličiny. Cíl práce byl splněn a byl navrhnut adaptabilní Neuro-regulátor, který zvládne dobře regulovat nelineární soustavu batyskafu v celém rozsahu regulované veličiny.

K návrhu Neuro-regulátoru bylo nejdříve nutné provést identifikaci soustavy batyskafu. Pro identifikaci byla zvolena diskrétní dynamická QNU s dynamikou 2. řádu, která dobře zvládla aproximaci reálných dat soustavy batyskafu. Výsledné hodnoty získané z identifikace soustavy bylo pak nutné podle matematického popisu použít pro diskrétní statickou QNU, která byla zapojena ve zpětné stavové vazbě jako samotný adaptabilní regulátor. Akční zásahy byly podpořeny zavedením proporciální složky.

Porovnáním průběhů regulace soustavy PID regulátorem a Neuro-regulátorem se jasně prokázala velice dobrá a rychlá regulace Neuro-regulátoru, zatímco regulace PID regulátorem byla buďto pomalá nebo dosahovala velkých překmitů a hlavně nereagovala na nelinearity soustavy.

Kromě dobré regulace má použití Neuro-regulátoru ještě další nesporné výhody. Zatímco PID regulátor se musí vždy na reálnou soustavu nastavit podle některé z teoretických metod nebo zdlouhavě experimentálně anebo kombinací obojího (například, není-li znám matematický popis soustavy), Neuro-regulátor pomocí identifikace soustavy provede nastavení sám a to i v případech, kdy není reálná soustava popsána žádným matematickým modelem. Pro Neuro-regulátor tedy stačí znát pouze vstupy a výstupy z reálné soustavy a soustavu jako takovou není třeba znát.

Pro Neuro-regulátor hovoří také fakt, že na schopnost dobré regulace neměla žádný (nebo velice zanedbatelný) vliv změna objemu vzduchové bublinky v batyskafu. S každou změnou objemu bublinky nadnášející batyskaf by bylo třeba znovu přeladit PID regulátor. Neuro-regulátor je tedy schopen pojmout kompletní dynamiku soustavy i

s dynamikou serva a dalších prvků, proto zde měla změna objemu vzduchu v batyskafu vliv jen na rychlost klesání nebo stoupání batyskafu.

Během experimentů bylo také prokázáno, že při regulaci soustavy batyskafu záleží více na velikosti změn žádané veličiny než na hloubce ponoru batyskafu. Hloubka ponoru batyskafu a jeho počáteční poloha mají však na průběh regulace také vliv. Tyto skutečnosti jsou způsobené nelinearitami soustavy a projevily se právě u regulace pomocí PID regulátoru, který s nimi měl velké problémy.

Dalším důležitým poznatkem bylo, že Neuro-regulátor při poklesu tlaku, tj. při simulaci poruchy v místě akční veličiny reagoval okamžitým spuštěním akčního členu, což poukazuje na jeho okamžitou reakci.

Možnost použití adaptabilního Neuro-regulátoru i pro další nelineární systémy je velice slibná. Výhodou je možnost jeho použití i pro neznámé reálné systémy.

Do budoucna by bylo vhodné se v této problematice dále zaměřit na velikosti změn žádané veličiny a ne pouze na žádanou hloubku nebo zpřesnění identifikace soustavy například identifikací v reálném čase.

Literatura

[1] Bíla, J.: Umělá inteligence a neuronové sítě v aplikacích. Skripta Fakulta strojní ČVUT Praha, 1995.

[2] Bukovsky, I.: Modeling of Complex Dynamic Systems by Nonconventional Artificial Neural Architectures and Adaptive Approach to Evaluation of Chaotic Time Series. Ph.D. Thesis, Faculty of Mechanical Engineering, CTU in Prague, 2007.

[3] Bukovsky, I., Bila, J.: „Basic Classification of Nonconventional Artificial Neural Units" (In Czech), *Proceedings of Seminar Nové Hrady*, Czech Technical University in Prague, FME, Czech Republic, 2007, pp. 76-80, ISBN: 978-80-01-03747-8

[4] Bukovsky, I., Hou, Z-G., Bila, J., Gupta, M., M.: *Foundation of Nonconventional Neural Units and their Classification*, International Journal of Cognitive Informatics and Natural Intelligence (IJCiNi), 2(4), October-December 2008, IGI Publishing, Hershey PA, USA, pp.29-43, ISSN 1557-3958.

[5] Wikipedia.org, http://en.wikipedia.org/wiki/Perceptron

[6] Bukovsky, I., Anderle, F., Smetana, L.,: "Quadratic Neural Unit for Adaptive Prediction of Transitions among Local Attractors of Lorenz System", accepted paper for 2008 *IEEE International Conference on Automation and Logistics*, Qingdao, China, 2008.

[7] Bukovsky, I., Smetana, L., Anderle, F.: "Adaptive Prediction of Transitions between Local Attractors of Lorenz's System". (in Czech), In: *New Methods and Approaches in the Fields of Control Technology, Automatic Control, and Informatics*, Czech Technical University, Prague, U12110.3, Czech Republic, 2008.

[8] Gupta, M., M., Liang J., and Homma N.: Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory, IEEE Press and Wiley-Interscience, published by John Wiley & Sons, Inc., 2003.

[9] Bukovsky, I., Bila, J.: „Basic Classification of Nonconventional Artificial Neural Units"(In Czech), *Proceedings of Seminar Nove Hrady*, Czech Technical University in Prague, FME, ISBN: 978-80-01-03747-8, Czech Republic, 2007, pp. 76-80.

[10] M. B. Kennel, R. Brown, and H. D. I. Abarbanel, *Determining embedding dimension for phase-space reconstruction using a geometrical construction*, Phys. Rev. A **45**, 3403 (1992).

[11] http://www.mpipks-dresden.mpg.de/~tisean/TISEAN_2.1/docs/chaospaper/node9.html, 13.11.2008, 3:46.

[12] Bukovsky I., S. Redlapalli and M., M. Gupta: "Quadratic and Cubic Neural Units for Identification and Fast State Feedback Control of Unknown Non-Linear Dynamic Systems", The Fourth International Symposium on Uncertainty Modeling and Analysis ISUMA 2003, IEEE Computer Society, Maryland USA, ISBN 0-7695-1997-0, 2003, pp.330-334.

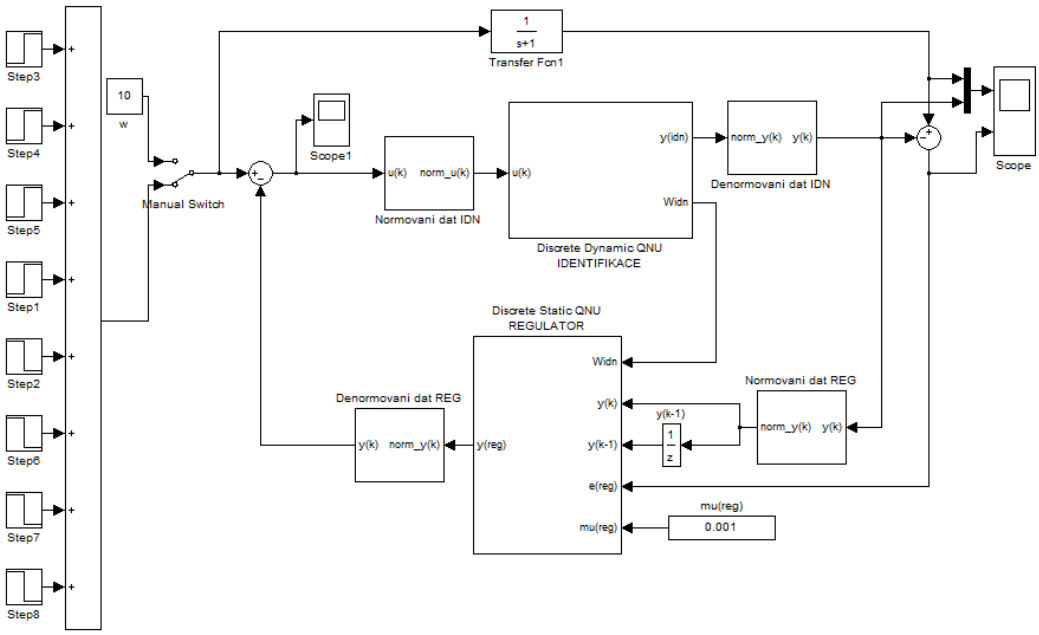
[13] Sborník odborného semináře: „Nové metody a postupy v oblasti přístrojové techniky, automatického řízení a informatiky", výukové středisko Herbertov 3.-5. Června 2005, Vyšší Brod.

[14] Demuth, H., Beale, M., Hagan, M.,: „Neural Network Toolbox for Matlab".

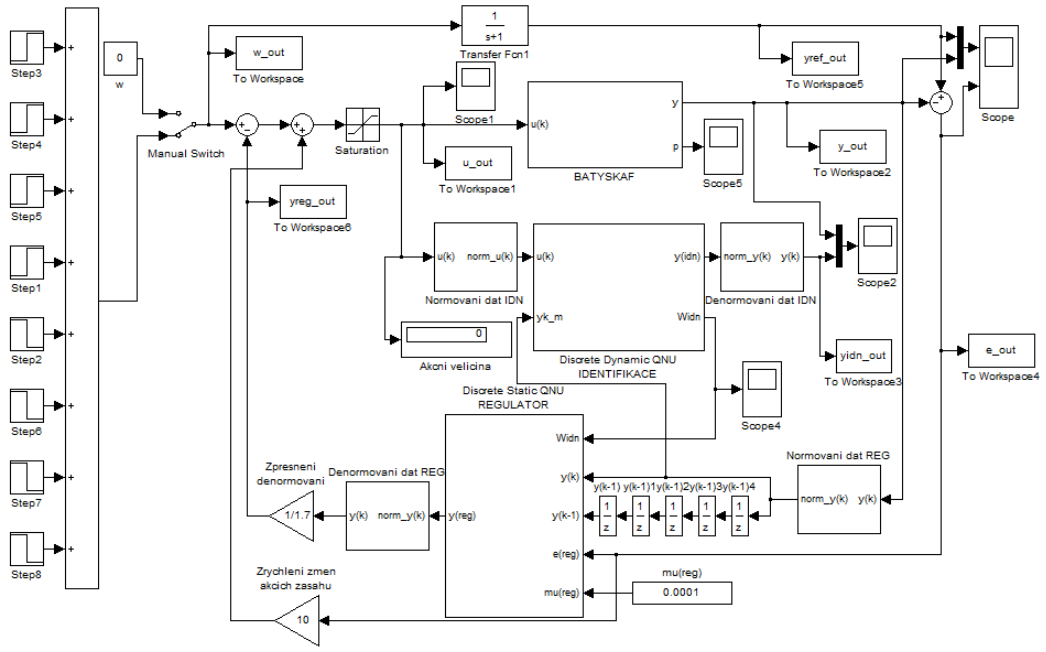
[15] http://www.fs.vsb.cz/books/NeuronoveSite/Aktiv_funkce.htm##F1, 6.12.2008, 13:00.

[16] http://www.ar-batyskaf.wz.cz/, 6.11.2008

Přílohy



Obr. 43 – Simulační schéma Neuro-regulátoru v Matlab/Simulink



Obr. 44 – Schéma neuro-regulátoru v Matlab/Simulink pro reáné řízení soustavy batyskaf

Výpis programu Batch-Trainingové metody Levenberg-Marquardt:

```
close all
clear all
clc;

load('datanorm.mat')

% nastaveni parametru
vzorkovani = 10;
kstart = 1;
%kstop = 5;
kstop = (length(ynorm)/vzorkovani)-2;
mi=0.01;
epochs=200;


uk = unorm(1:length(ynorm)-2);
yk = ynorm(1:length(ynorm)-2);
wk = w(1:length(y)-2);
yk1 = ynorm(2:length(ynorm)-1);
yk2 = ynorm(3:length(ynorm));

uk_s = uk(1:vzorkovani:length(uk));
yk_s = yk(1:vzorkovani:length(yk));
wk_s = wk(1:vzorkovani:length(wk));
yk1_s = yk1(1:vzorkovani:length(yk1));
yk2_s = yk2(1:vzorkovani:length(yk2));

x0=ones(length(yk2_s),1);

x=[x0 uk_s yk_s yk1_s];

n=size(x,2);

W=(rand(n)-.5)*0.1;
W0=W;

delka=length(yk2_s);

wah=0;
for i=1:n
for j=i:n
    wah=wah+1;
end
end

J=zeros(delka,wah);

uspesnychepoch=0;

for epoch=1:epochs

for k=1:delka
yn(k)=x(k,:)*W*x(k,:)';

waha=0;
for i=1:n
```

```
for j=i:n
    waha=waha+1;
    J(k,waha)=(x(k,i)*x(k,j));
end
end
end

e=yk2_s-yn';

JJ=e'*e;

dw=((J'*J+1/mi*eye(wah))^-1)*J'*e;

Wpredch=W;

waha=0;
  for i=1:n
    for j=i:n
        waha=waha+1;
        Wnasl(i,j)=W(i,j)+dw(waha);
    end
  end

  for k=1:delka
    ynnasl(k)=x(k,:)*Wnasl*x(k,:)';
  end
e=yk2_s-ynnasl';
JJnasl=e'*e;



    if JJnasl<JJ
     uspesnychepoch=uspesnychepoch+1;
     W=Wnasl;
     J_p(uspesnychepoch) = JJ;
     W_p(:,:,uspesnychepoch)=W;
     else
     mi=mi/1.2
    end

end

figure
hold on
grid on
plot(-yn,'g','LineWidth',3)
plot(-yk2_s,'r','LineWidth',2)
title('Levenberg-Marquardt algoritmus, prubeh yr a yn')
legend('yn - DiscStatQNU','yr - Batyskaf')
xlabel('k [samples]'); ylabel('yn, yr [cm]');

figure
hold on
grid on
plot(yk2_s-yn','r')
title('Levenberg-Marquardt algoritmus, prubeh chyby')
xlabel('k [samples]'); ylabel('err [cm]');
```

```
w00(1)=W0(1,1);
w01(1)=W0(1,2);
w02(1)=W0(1,3);
w03(1)=W0(1,4);
w10(1)=W0(2,1);
w11(1)=W0(2,2);
w12(1)=W0(2,3);
w13(1)=W0(2,4);
w20(1)=W0(3,1);
w21(1)=W0(3,2);
w22(1)=W0(3,3);
w23(1)=W0(3,4);
w30(1)=W0(4,1);
w31(1)=W0(4,2);
w32(1)=W0(4,3);
w33(1)=W0(4,4);
for epocha=2:uspesnychepoch
w00(epocha)=W_p(1,1,epocha);
w01(epocha)=W_p(1,2,epocha);
w02(epocha)=W_p(1,3,epocha);
w03(epocha)=W_p(1,4,epocha);
%w10(epocha)=W_p(2,1,epocha);
w11(epocha)=W_p(2,2,epocha);
w12(epocha)=W_p(2,3,epocha);
w13(epocha)=W_p(2,4,epocha);
%w20(epocha)=W_p(3,1,epocha);
%w21(epocha)=W_p(3,2,epocha);
w22(epocha)=W_p(3,3,epocha);
w23(epocha)=W_p(3,4,epocha);
%w30(epocha)=W_p(4,1,epocha);
%w31(epocha)=W_p(4,2,epocha);
%w32(epocha)=W_p(4,3,epocha);
w33(epocha)=W_p(4,4,epocha);
end

figure
hold on
grid on
plot(w00,'b')
plot(w01,'b:')
plot(w02,'b-.')
plot(w03,'g')
%plot(w10,'g:')
plot(w11,'g-.')
plot(w12,'r')
plot(w13,'r:')
%plot(w20,'r-.')
%plot(w21,'c')
plot(w22,'c:')
plot(w23,'c-.')
%plot(w30,'m')
%plot(w31,'m:')
%plot(w32,'m-.')
plot(w33,'k')
legend('w00','w01','w02','w03','w11','w12','w13','w22','w23','w33')
%legend('w00','w01','w02','w03','w10','w11','w12','w13','w20','w21','w
22','w23','w30','w31','w32','w33')
title('Levenberg-Marquardt algoritmus, prubeh neuralnich vah')
xlabel('pocet epoch (cyklu uceni)'); ylabel('w(i,j) [-]');
```