|  |
| --- |
| Předmět – Neuronové síťě |
| Ročníková práce |
| Řízení laboratorního modelu G.U.N.T. RT 050 |

|  |
| --- |
| Ing. Mariška Martin  15.6.2013 |

Obsah

# Seznam symbolů a zkratek

DIC Direct inverse control

IMC Internal model kontrol

MSE Mean Square Error

# Úvod

Ročníková práce má dva hlavní cíle. Za prvé se má namodelovat optimální dynamický neuronový model laboratorního modelu motoru GUNT RT 050. Za druhé pomocí dvou metod řízení automaticky řídit otáčky modelu motoru. Požadované metody řízení jsou:

* Přímé inverzní řízení pomocí inverzní neuronové sítě (DIC – Direct Inverse Control)
* Řízení s vnitřním modelem pomocí inverzní neuronové sítě a dynamického neuronového modelu soustavy (IMC – Internal Model Control)

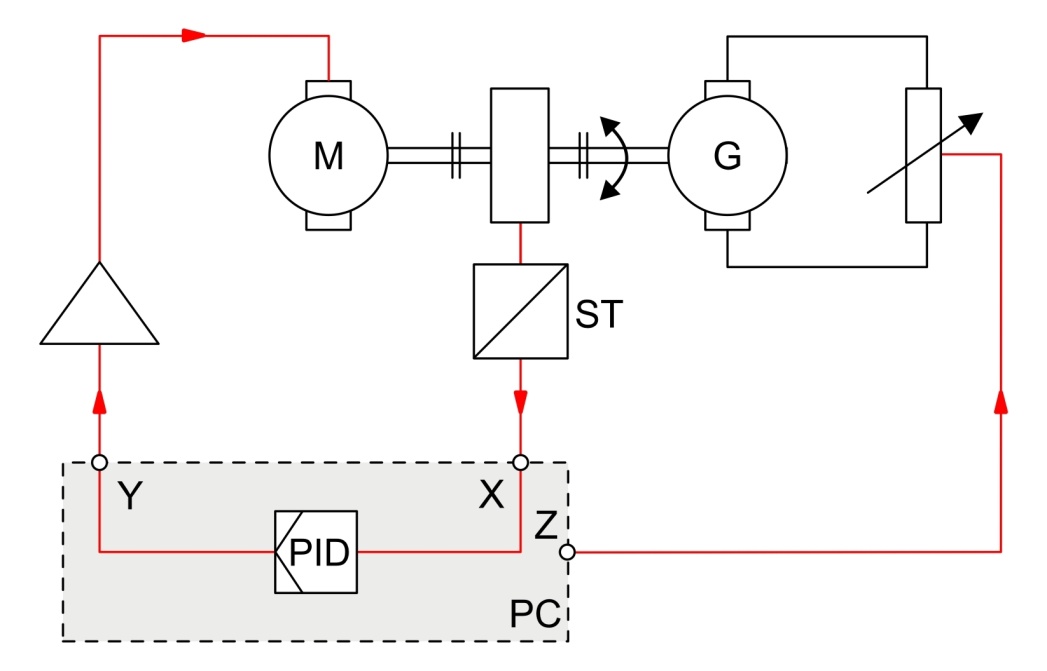
Jednotlivé metody řízení jsou popsány a vysvětleny například v publikacích [1], [2]. Způsob tvorby dynamického neuronového modelu je popsán v publikaci [1].

# Obecné informace o modelu a měření

Laboratorní vybavení GUNT RT 050 má určeny pro řízení počítačem vstupní napětí v rozmezí 0V až 5V a na výstupu ze snímače otáček je výstupní napětí v rozmezí 0V až 10V. Pro jeho řízení je v prostředí Simulink vytvořen a předem připraven model a dynamická knihovna. Tento předpřipravený model byl použit, jak pro sběr dat, tak i pro následné řízení soustavy.



Obrázek – Reálný model motoru G.U.N.T. RT 050 [zdroj: www.gunt.de]



Obrázek – Schéma zapojení s PC udávané pro model G.U.N.T RT 050 [zdroj: www.gunt.de]

Měření výstupních a vstupních parametrů je, již na úrovni modulu v simulinku, normalizováno na interval <-1, 1>. Proto se v celý projekt zmiňuje pouze o rozsazích -1, 1, tedy převedený vstupní rozsah napětí z <0, 5> V a výstupní rozsah napětí <0, 10> V. Oba jsou převedeny pomocí metody zvané „Range Normalization“ (rozsahová normalizace). Ta umožňuje určitý rozsah hodnot mapovat na jiný rozsah hodnot.

Rozsahová normalizace:

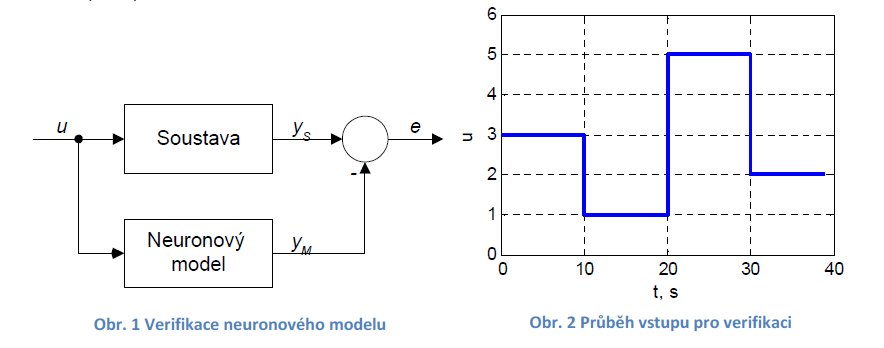
Rozsahová denormalizace:

Kde , je normalizovaná maximální, minimální hodnota a , je maximální, minimální hodnota původního rozsahu. Více v [3].

# Optimální dynamický neuronový model soustavy

## Zadání

Navrhněte optimální dynamický neuronový model soustavy pro řízení otáček G.U.N.T. RT 050. Neuvažujte zátěž motoru. Kritérium optimality modelu definujte a jeho definici vysvětlete. Verifikaci neuronového modelu proveďte podle zapojení na obr. 1 pro průběh vstupu u podle obr. 2.



## Kritérium optimality

Definice optimality sítě je daná touto kriteriální funkcí:

kde je skutečné výstupní napětí, je výstupní napětí stanovené neuronovou sítí, je počet vzorků pro test a je celkový počet neuronů v použité neuronové síti.

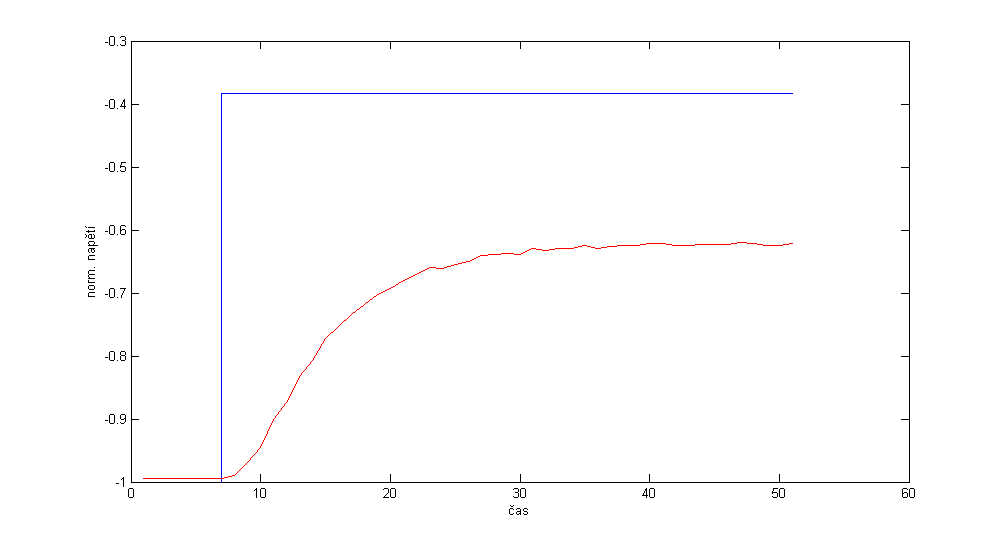
Cílem optimalizace, podle tohoto kritéria, je tedy co nejpřesnější odhad výstupních hodnot vůči reálnému průběhu výstupu (první část kriteriálního vzorce) a zároveň minimalizujeme počet neuronů obsažených v  neuronové síti (zachováme tím i vlastnost generalizace sítě). Kriteriální funkci budeme tedy minimalizovat. Vhodnější model má menší hodnotu kritéria.

## Identifikace a volba vzorkovací frekvence

Prvním voleným parametrem pro nasnímání dat, které budou dále zpracovávány, je vzorkovací perioda a předpoklad jakého řádu je zkoumaný systém. Volba těchto dvou parametrů ovlivňuje následný model systému. V rámci konzultace ohledně snímání dat, bylo doporučeno, aby počet nasnímaných dat byl mezi 15-20 vzorky (informace je odvozena ze zkušenosti) na přechod soustavy znovu do ustáleného stavu. Z přiloženého obrázku níže, je vidět odezva na definovaný skok vstupního napětí.

Doba od začátku reakce až do ustálení trvá zhruba 15 sekund (odečteno z grafu viz. Obrázek 3). Tedy pro splnění požadavku 20 hodnot na skok je výpočet vzorkovací periody:

Dále řád soustavy je také odvozen z grafu na obrázku 3. U soustav druhého a vyššího řádu je známé, že se na začátku grafu objeví doba průtahu. Existence doby průtahu indikuje systém minimálně druhého řádu, proto pro rovnice a trénování modelu je jako výchozí řád soustavy nejprve předpokládán druhý řád soustavy.



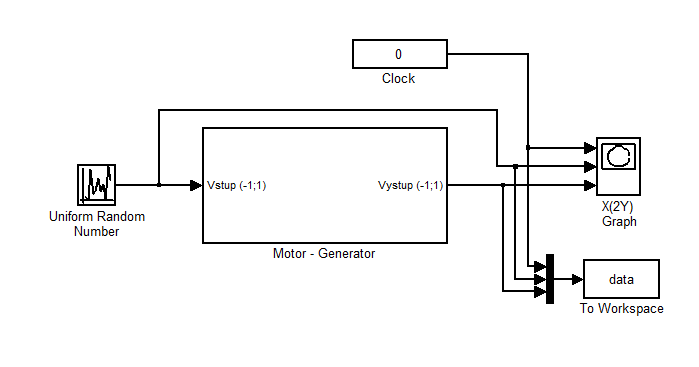
Obrázek – reakce systému na vstupní skok

## Nalezení optimálního modelu

Cílem této kapitoly je popsat postup nalezení optimálního neuronového modelu, kde optimality je definována kriteriální funkcí.

### Naměřená množina dat

Celá množina naměřených dat se měřila cca 60 minut. A zapojený model měření v simulinku vypadal následovně.



Obrázek – Simulink schéma pro měření dat reálného systému

Průbě celé naměřené množiny je zobrazen níže, aby bylo vizuálně kontrolovatelné, že byl proměřen vhodně celý pracovní prostor systému. Volba vstupního signálu byl volen náhodně pomocí bloku „Uniform Random Number“ (seed = 615482, min = -1, max = 1).

measure075.emf

Obrázek – Celá naměřená množina dat

measure075-detail.emf

Obrázek – Detail začátku naměřené množiny dat

### Trénování neuronové sítě

Trénování neuronové sítě proběhlo na vybrané podmnožině naměřených dat. Množina vypadala následovně viz. Obrázek 7 a měla 600 hodnot.

C:\Users\MAR\Desktop\trainSet.emf

Obrázek – Trénovací množina

Pro nalezení optimálního modelu byl zhotoven skript, který hledal vhodnou topologii a opakoval stejný scénář trénování několikrát po sobě (s jinými počátečními podmínkami pro neuronovou síť), aby bylo možné zobrazit i průměrné hodnoty kritéria. Ve výsledcích bude zobrazen krabicový graf, který vyjadřuje nejvhodnější topologii pro tento systém.

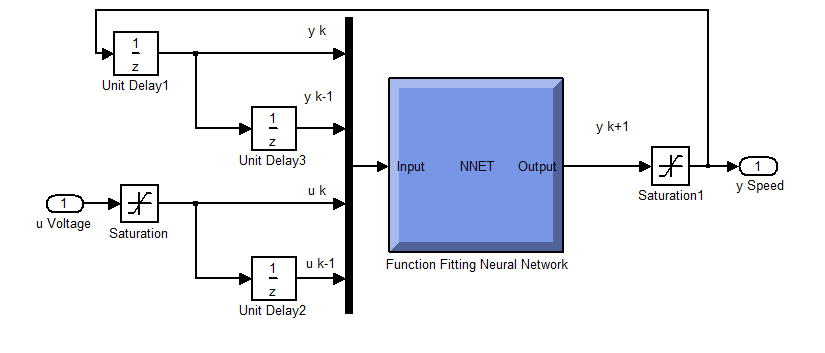
Algoritmus hledání zajišťuje uložení dat i s neuronovou sítí. Skript ukládá data, pouze pokud nalezne model s lepším výsledkem kriteriální funkce. Skript je přiložen v příloze A.

Pro trénování neuronové sítě byla sestavena vstupní množina dat, vycházející z diferenciální rovnice pro soustavu druhého řádu. Víc informací o diskrétním popisu soustav v [1]. Obecná rovnice pro soustavu druhého řádu vypadá následovně.

Z tohoto vyplívá, že cíle trénování budou hodnoty na pozici a vstupy pro trénování budou hodnoty v uvedeném pořadí. Následuje skript pro vytvoření vstupů a cílů pro trénování z výchozí naměřené množiny dat.

% pole u, y jsou výchozí naměřené vektory dat  
% vytvorit vstupy / inputs  
inputs = [y(2:end-1); y(1:end-2); u(2:end-1); u(1:end-2)];  
% vytvorit cíle / targets  
targets = y(3:end);

Pro zjednodušení používání neuronového modelu v simulinku, je vytvořen blok subsystému. Schéma vnitřního zapojení je na obrázku níže. Tento blok se pak používá dále ve schématech pro řízení.



Obrázek – Schéma vnitřního zapojení bloku pro neuronový model

### Výsledky

Tabulka s přehledem topologií a nejlepších hodnot kriterií z 10 replikací pro danou topologii.

|  |  |
| --- | --- |
| **Tabulka 1 – Porovnání hodnot kritérií pro různé topologie** | |
| *Topologie* | *Nejmenší hodnota kritéria (z 10 replikací)* |
| 4 – 5 – 1 | -4.4056 |
| 4 – 10 – 1 | -4.5718 |
| 4 – 13 – 1 | -4.5880 |
| 4 – 15 – 1 | -4.5758 |
| 4 – 4 – 3 – 1 | -4.5390 |
| 4 – 5 – 5 – 1 | -4.5232 |
| 4 – 7 – 7 – 1 | -4.6122 |
| 4 – 8 – 1 | -4.5085 |

Krabicový graf s hodnotami kriteria pro jednotlivé topologie. Pokus byl proveden pro každou topologii v 10 replikacích (pokus se stejným scénářem) a ve spodní části je topologie označena počtem neuronů ve skryté vrstvě.

boxPlot.emf

Obrázek – Krabicový graf kriterií podle typu topologie

Z krabicového grafu plyne, že nejvhodnější topologie je s 10 skrytými neurony v jedné skryté vrstvě. Průměrně poskytuje nejnižší hodnoty kriteriální funkce.

## Problémy a doporučení

V průběhu pokusů a testování neuronové sítě jsem došel k závěru (který je ověřitelný měřením), že pokud se trénuje dynamický model z mnoha naměřených vzorků (2000 vzorků a více), tak je mnohem obtížnější jednak natrénovat neuronovou síť, ale výsledný model je i přeučený a není snadné ho následně dobře řídit. Pro některé hladiny žádané veličiny vracel neuronový model stále malinko odlišné veličiny a tím se stále projevovalo mírné kolísání kolem požadované hladiny.

## Verifikace dynamického modelu

Cílem verifikace dynamického neuronového modelu, je porovnat průběhy modelu s reálnou soustavou. Průběh žádané veličiny je uveden v zadání. A jednotlivé naměřené hodnoty jsou vyneseny do grafu na obrázku níže.

porovnani.emf

Obrázek – Porovnání průběhu neuronového modelu s reálnou soustavou

Jako číselné vyjádření kvality je ještě uvedena hodnota MSE (často se používá pro porovnávání i v jiných publikacích).

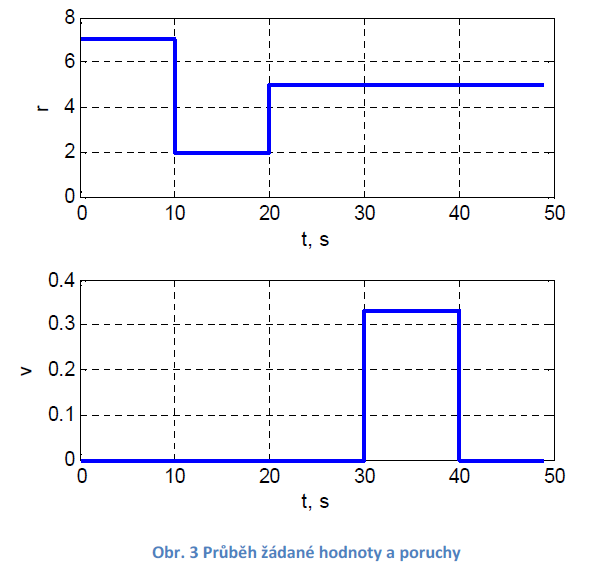
# Automatické řízení otáček soustavy

## Zadaní

Navrhněte následující způsoby automatického řízení soustavy pro řízení otáček G.U.N.T. RT 050.

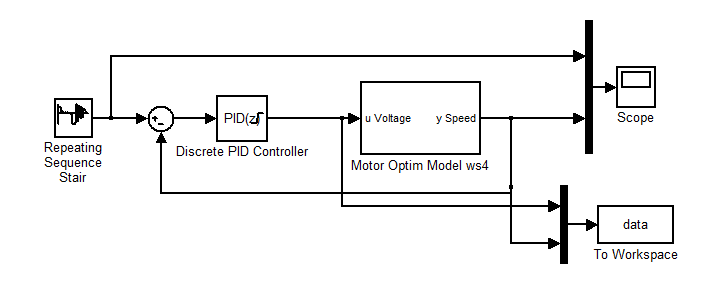
1. Přímé inverzní řízení pomocí inverzní neuronové sítě
2. Řízení s vnitřním modelem pomocí inverzní neuronové sítě a dynamického neuronového modelu soustavy

Výsledné regulátory otestujte pro průběh žádané hodnoty regulované veličiny *r* a poruchy *v* (zátěž motoru) uvedený na obr. 3. Dosažené regulační pochody porovnejte se simulacemi dosaženými pomocí neuronového modelu a výsledky diskutujte.



## Řízení metodou DIC

Obecný popis a východiska pro metodu přímého inverzního řízení (DIC) je například v publikaci [1], [2]. Pro natrénování inverzního modelu soustavy byla použita data z originální soustavy řízené pomocí jednoduchého diskrétního PI regulátoru (P = 0.2, I = 0.2) s předcházením tzv. „wind-up“ efektu za pomocí metody „back-calculation“. Sesbírány byly tedy hodnoty vstupu až na výstupu PI regulátoru a jako výstupní hodnoty se ukládali data na výstupu ze soustavy. Pro názornější představu je zobrazen schéma zapojení ze simulinku.



Obrázek – Simulink schéma měření dat pro inverzi

trenovaciDataInverze.emf

Obrázek – Naměřená data pro trénování inverze

Pro trénování inverze byl zvolen stejný postup jako u hledání optimálního dynamického neuronového modelu. Struktura neuronového modelu byla zvolena na začátku jako nejjednodušší inverze soustavy 1. Řádu. Přesná teoretická východiska jsou v [1]. Tedy rovnice pro trénování vypadá takto:

Kde na levé straně rovnice jsou cíle pro trénování tedy . Funkce představuje inverzní funkci, kterou se má aproximovat neuronová síť a vstupy pro trénování jsou .

% pole u, y jsou výchozí naměřené vektory dat  
% vytvorit vstupy / inputs  
inputs = [y(2:end); y(1:end-1)];  
% vytvorit cíle / targets  
targets = u(1:end-1);

Dále byl proveden i pokus s inverzí nad soustavou druhého řádu, ale významnou změnu to pro řízení nepřineslo. Pokud se vycházelo ze soustavy druhého řádu, tak naopak bylo těžší natrénovat neuronovou síť s dostatečnou přesností a následně byl regulační pochod náchylnější na menší změny (co může v realitě způsobovat i rušení a znatelným způsobem ovlivňovat regulační pochod).

Následně bylo provedeno hledání optimálního inverzního modelu a nejlepší kriteriální hodnotu měla topologie 2 – 5 – 5 – 1 s hodnotou kritéria -2.6765. Pro názornost je na obrázku níže zobrazeno porovnání originálního a aproximovaného průběhu neuronovou sítí a krabicový graf pro porovnání vlivu různých topologií na trénování.

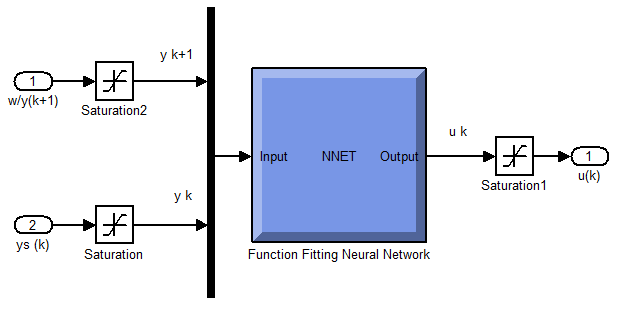
boxPlot3.emf

Obrázek – Krabicový graf pro porovnání hodnot kritéria pro různé topologie

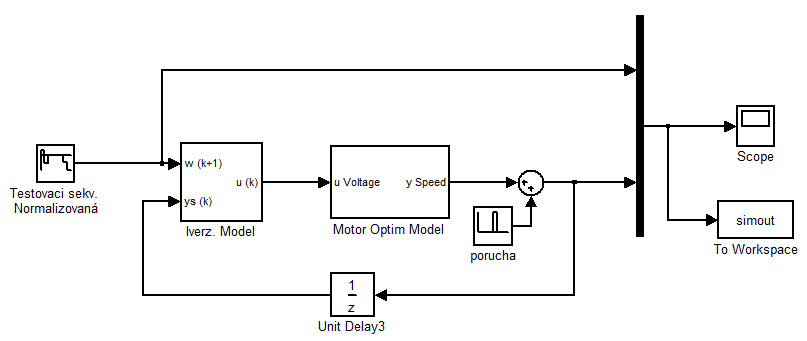
criteriaEval.emf

Obrázek – Porovnání průběhu ureal (pro reálný systém) a uinn (pro výstup z inverzní sítě)

Pro řízení metodou DIC se zapojuje inverzní model předřazením před řízenou soustavu, více informací o metodě DIC a zapojení v [1], [2]. Následuje nejprve schéma zapojení subsystému inverzního neuronového modelu a následně schéma zapojení celé soustavy pro experimenty v prostředí simulink.



Obrázek – Schéma zapojení bloku susbsytému s inverzním modelem



Obrázek – Schéma zapojení regulace pomocí metody DIC

Podle zadání bylo řízení provedeno řízení modelu a soustavy na testovacím průběhu definovaném v zadání.

dic2.emf

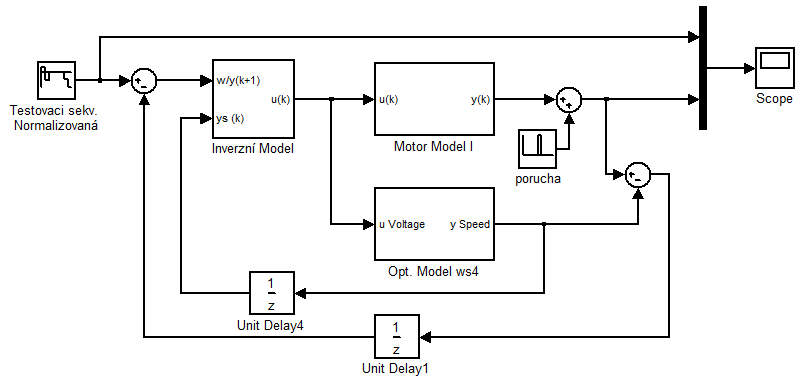
Obrázek – Test řízení metodou DIC na testovacím průběhu s neuronovým modelem

dicReal1.emf

Obrázek – Test řízení metodou DIC na testovacím průběhu s reálnou soustavou

## Řízení metodou IMC

Obecný popis a teoretická východiska pro metodu řízení pomocí vnitřního modelu (IMC) je například v publikaci [1], [2]. Pro natrénování inverzního modelu soustavy byl použit stejný postup jako u metody DIC v předešlé kapitole. Metoda IMC se z pohledu realizace liší od DIC pouze zapojením.



Obrázek – Schéma zapojení bloků v simulinku pro metodu IMC

Byl nejprve proveden test regulace na testovací sekvenci žádané veličiny pro řízení neuronového modelu za pomocí jiného neuronového modelu. Byl znovu natrénován, aby nevracel identická data. Až po tomto testu bylo nasazeno schéma na reálnou soustavu a blok „motor model“ byl nahrazen blokem pro práci s reálnou soustavou. Níže jsou uvedeny oba průběhy, jednou s neuronovým modelem a jednou pro řízení reálné soustavy.

imc2.emf

Obrázek – Test řízení metodou IMC na testovacím průběhu s neuronovým modelem

imcReal1.emf

Obrázek – Test řízení metodou IMC na testovacím průběhu s reálnou soustavou

# Porovnání a diskuse výsledků metod řízení

V této kapitole budou porovnány obě zadané metody řízení. Porovnávány budou z pohledu řízení, tedy z pohledu reakce na změnu žádané veličiny a reakce na poruchy. Kvalita obou metod bude vyjádřena kvantitativně pomocí MSE (mean square error).

Byl proveden experiment a naměřeny hodnoty regulačních pochodů pro obě metody s reálnou soustavou. Hodnoty jsou zobrazeny v grafu na obrázku níže.

porovnaniDicImc.emf

Obrázek – Porovnání obou metod řízení (DIC s IMC) na reálné soustavě

Hodnota MSE číselně vyjadřuje, který regulační pochod se více přibližuje k žádané veličině (menší hodnota MSE znamená přesnější regulační pochod).

Kde a jsou příslušné vektory naměřených hodnot. Pro výpočet byla použita funkce poskytnutá prostředím Matlab. Funkce má název *mse* a používá se stejným způsobem, jako je uveden výpočet výše. Z číselného porovnání lze říci, že přesnější regulační pochod poskytuje metoda IMC.

Největší výhodou metody IMC oproti DIC, je zpětná vazba a tedy již teoretického hlediska je IMC schopna na základě zpětné vazby upravit akční veličinu tak, aby se případný poruchový signál anuloval. Poruchový signál je pro metodu DIC případem, kdy se náhle změní parametry soustavy a regulátor se o tom nedozví. Z principu lze u metody DIC předpokládat velkou náchylnost na poruchové signály. Druhé hledisko je, že metoda IMC vyžaduje jak inverzní model soustavy, tak i normální dynamický model soustavy. Je tedy mnohem náročnější na realizaci. U obou metod je obecným problémem vytvoření inverzního modelu soustavy, protože u některých systémů nemusí být jednoduché jeho natrénování. Způsob jakým získat lepší množinu dat pro trénování je například danou soustavu řídit pomocí vhodné varianty diskrétního PID regulátoru. Tento způsob byl použit i v této práci pro trénování inverzního modelu soustavy.

# Závěr

Byl zhotoven optimální neuronový model laboratorního systému GUNT RT 050, realizovány a otestovány dvě metody řízení (DIC, IMC). Vhodnější metodou pro řízení soustavy je metoda IMC, protože je méně náchylná na poruchové signály a vhodně upravuje regulaci díky zpětné vazbě. V teoretické i praktickém porovnání byla lepší metoda IMC.

V průběhu práce byl nalezen problém, že při velkém množství vzorků, pro trénování neuronového modelu systému a pro trénování inverze, je obtížné dobře natrénovat neuronovou síť, tak aby dostatečně aproximovala reálný systém a nebyla přeučená. Při velkém počtu vzorků pravděpodobně vzorky obsahují spoustu zašuměných informací. V průběhu trénování a testování se tedy ukázalo, že je vhodnější brát množiny o 600 až 1500 vzorcích. Vzorky obsahovali zhruba 5 až 15 změn vstupní veličiny. Perioda mezi změnami byla volena přibližně tak, aby se soustava stihla na výstupu ustálit.

# Literatura

[1] DOLEŽEL, Petr. „Umělé neuronové sítě v modelování a řízení kontinuálního bioreaktoru“. Diplomová práce. Univerzita Pardubice. 2008.

[2] Direct Inverse Control & Internal Model Control. In: [online]. [cit. 2013-06-15]. Dostupné z: https://courses.cs.ut.ee/2008/modelling-and-control/slides/dic-and-imc.pdf

[3] Range Normalization. In: [online]. [cit. 2013-06-15]. Dostupné z: http://www.heatonresearch.com/wiki/Range\_Normalization

# Přílohy

Příloha A – Optimalizační skript

% script for discover optimal dynamic neural model of motor  
  
% keep for all test value of criteria, performance, ...  
% for each topology do 5 tests  
% save better model tagged with criteria value..  
  
inputs = inputs;  
targets = targets;  
startTime = now();  
topologyPatterns = {5; 8; 10; 13; 15; [4 3];[5 5]; [7 7]}; % 15; 20; [8 8]; [5 5] [11 15 18 20 25]; %[11 15 18 20 22 25 29];  
replications = 10;  
maxTrainTimeInSec = 1\*60;  
count = 1;  
optimizedTopology = cell(replications\*length(topologyPatterns),1);  
for k = 1:length(topologyPatterns)  
 for j = 1:replications  
 optimizedTopology{count} = topologyPatterns{k};  
 count = count + 1;  
 end  
end  
criterias = ones(length(optimizedTopology), 1).\*99;  
optimizeData = cell(length(optimizedTopology), 2);  
%vyrobit slozku pro vysledky  
folderName = ['vysledky\_' datestr(startTime,'yyyy-mm-dd\_HHMMss')];  
mkdir(folderName);  
  
%vlastni cyklus pro uceni a ukladani nejlepsich vysledku  
for k = 1:length(optimizedTopology)  
 [net, trainResults] = learnNet(inputs, targets, optimizedTopology{k}, maxTrainTimeInSec);  
 crit = Criteria(net, inputs, targets);  
 critValue = crit.getValue(); %vypocet kriteria [minimalizujeme]  
 simplefitOutputs = net(inputs); %simulace hodnot  
 actPerformance = perform(net, targets, simplefitOutputs);  
 disp([num2str(k) '. top/perf/criteria [' num2str(optimizedTopology{k}) ', ' num2str(actPerformance) ', ' num2str(critValue) ']']);  
 if(critValue < min(criterias))  
 save([folderName '/workspace\_' num2str(critValue) '.mat']);  
 disp('> model saved...');  
 end  
 criterias(k) = critValue;  
 optimizeData{k,1} = num2str(optimizedTopology{k});  
 optimizeData{k,2} = actPerformance;  
 optimizeData{k,3} = critValue;  
 optimizeData{k,4} = trainResults.perf;  
 save([folderName '/measureInfo'], 'optimizeData');  
 disp('> info data saved...');  
end  
[~, idx] = min(criterias);  
disp(['best model is at ' num2str(idx)]);

function [net, trainResults] = learnNet(inputs, targets, topology, maxTrainTime)  
% povinne jsou prvni dva argumenty "inputs, targets"  
  
clear('net');  
net = fitnet(topology);  
% Setup Division of Data for Training, Validation, Testing  
net.divideParam.trainRatio = 80/100;  
net.divideParam.valRatio = 10/100;  
net.divideParam.testRatio = 10/100;  
% nezobrazovat okno uceni  
net.trainParam.showWindow = false;  
  
net.trainParam.time = maxTrainTime;  
net.trainParam.epochs = 1000;  
net.trainParam.max\_fail = 100;  
  
% Train the Network  
[net, trainResults] = train(net, inputs, targets);