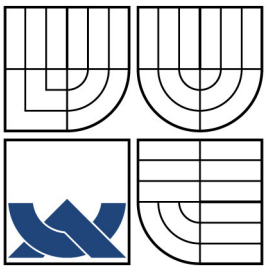
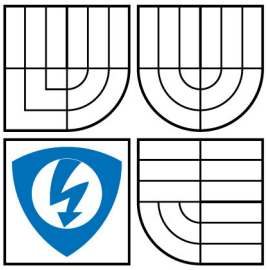


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

ADAPTIVNÍ OPTIMÁLNÍ REGULÁTORY S PRINCIPY UMĚLÉ INTELIGENCE V PROSTŘEDÍ MATLAB - B&R

ADAPTIVE OPTIMAL CONTROLLERS WITH PRINCIPLES OF ARTIFICIAL INTELLIGENCE

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — Bc. MICHAL MRÁZEK
AUTHOR

VEDOUCÍ PRÁCE — prof. Ing. PETR PIVOŇKA, CSc.
SUPERVISOR

BRNO 2008

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ
Fakulta elektrotechniky
a komunikačních technologií
Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student: Mrázek Michal, Bc. ***ID:*** 89948
Ročník: 2 ***Akademický rok:*** 2007/08

NÁZEV TÉMATU:

Adaptivní optimální regulátory s principy umělé inteligence v prostředí MATLAB - B&R

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s metodikou návrhu adaptivních optimálních regulátorů. Seznamte se s použitím klasických identifikačních metod a identifikačních metod na bázi neuronových sítí. Zaměřte se na možnosti implementace adaptivního optimálního regulátoru z prostředí programu MATLAB do programovatelného automatu B&R. V identifikačním algoritmu realizujte grafické vyhodnocení průběhů identifikace. Ověřte a porovnejte adaptivní optimální regulátor s pevně nastaveným PID regulátorem ve spojení program MATLAB - program. automat B&R - simulační a fyzikální modely, zejména s ohledem na změnu dynamiky modelů. Porovnejte vlastnosti klasické identifikace a identifikačních algoritmů na bázi neuronových sítí. Implementujte adaptivní optimální regulátor do programovatelného automatu.

DOPORUČENÁ LITERATURA:

PIVOŇKA, P.:Optimalizace regulátorů. VUT Brno, skriptum, 2005.
PIVOŇKA, P.: Číslicová řídicí technika, VUT Brno, skriptum, 2003
BOBÁL, V. a kol.: Praktické aspekty samočinně se nastavujících regulátorů. VUTIUM, Brno, 1999.

Termín zadání: 3.12.2007 ***Termín odevzdání:*** 26.5.2008

Vedoucí projektu: prof. Ing. Petr Pivoňka, CSc.

prof. Ing. Pavel Jura, CSc.
předseda oborové rady

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ Fakulta elektrotechniky a komunikačních technologií Ústav automatizace a měřicí techniky

UPOZORNĚNÍ:

Abstrakt

Diplomová práce popisuje návrh adaptivního optimálního regulátoru, který přizpůsobuje parametry algoritmu na základě průběžných informací o soustavě s ohledem na zvolené kritérium. V práci jsou popisovány některé metody identifikace soustavy, která je důležitou součástí adaptivního řízení. Dále je v prostředí MATLAB/Simulink a GUI (Graphical User Interface) porovnáván adaptivní optimální regulátor s diskrétním ekvivalentem PID regulátoru (resp. PSD regulátoru). Výsledný algoritmus je ověřen v simulacích s reálným systémem. Komunikaci PC s reálným systémem zajišťuje programovatelný automat B&R.

Klíčová slova

adaptivní optimální řízení, LQ regulátor, identifikace, metoda nejmenších čtverců, neuronová síť, metoda Back-propagation, metoda Levenberg-Marquardt, samočinně se nastavující regulátor, PSD regulátor, S-PD regulátor, programovatelný automat, PLC

Abstract

Master’s thesis describes adaptive optimal controller design which change parameters of algorithm based on the system information regard for optimal criterion. In this thesis are described some methods of identification, which are important part of a self-adapting control. Adaptive optimal controller and discrete equivalent PID (let us say PSD) are compared in MATLAB/Simulink and GUI (Graphical User Interface). Resulting algorithm is verified by simulation on the real system. Communication with the real system is supported by programmable logic controller B&R.

Key words

adaptive optimal control, LQ controller, identification, method of recursive last squares, neural network, Back-propagation method, Levenberg-Marquardt method, self-tuning controller, PSD controller, S-PD controller, programmable logic controller, PLC

MRÁZEK, M. *Adaptivní optimální regulátory s principy umělé inteligence v prostředí MATLAB - B&R.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 73 s. Vedoucí diplomové práce prof. Ing. Petr Pivoňka, CSc.

P r o h l á š e n í

„Prohlašuji, že svou diplomovou práci na téma "Adaptivní optimální regulátory s principy umělé inteligence v prostředí MATLAB – B&R" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne : Podpis:

Děkuji prof. Ing. Petru Pivoňkovi, CSc. za cenné připomínky a rady při vypracování diplomové práce.

OBSAH

SEZNAM OBRÁZKŮ

1. ÚVOD

Adaptivní optimální regulátory přizpůsobují parametry algoritmu na základě průběžných informací o soustavě s ohledem na kritérium bez nutnosti zásahu uživatele. Pokud se jedná o soustavu, která nemění svoje parametry, stačí jednorázová identifikace před vlastním řízením. Pokud ale soustava mění v čase svoje parametry, je nutné provádět průběžnou identifikaci.

Identifikace může být realizována různými metodami (např. metodou nejmenších čtverců). Výsledné hodnoty parametrů modelu regulované soustavy jsou použity pro výpočet nového akčního zásahu.

Práce se zabývá metodikou návrhu adaptivního optimálního regulátoru, typy klasických identifikačních metod nebo metod založených na neuronové síti. Nejprve byl vytvořen algoritmus obecného optimálního regulátoru, který se snaží dostat všechny stavy do nuly. Následně byl výpočetní algoritmus upraven pro sledování žádané hodnoty a dosažení nulové ustálené odchylky.

Dále byl do výpočtu akčního zásahu optimálního regulátoru vložen algoritmus identifikace soustavy, a to nejprve metoda nejmenších čtverců a potom metoda založená na neuronové síti (Levenberg-Marquardt).

Součástí práce je porovnávání optimálního řízení a řízení pomocí pevně nastavených diskrétních ekvivalentů PID regulátorů. Výsledky byly ověřeny na fyzikálním modelu. Komunikaci (MATLAB/Simulink – fyzikální model) zajišťoval programovatelný automat B&R. V práci je také ukázáno, jak se změní průběhy signálu v řízení při změně dynamiky soustavy. Byl vytvořen program, který demonstruje adaptivní optimální regulaci zvolené soustavy.

2. MODELY A IDENTIFIKACE PROCESU

V praxi se velmi často vyskytují regulátory s pevně nastavenými parametry pro danou soustavu. V těchto případech není nutné soustavu průběžně identifikovat. Může ale nastat případ, kdy se v průběhu regulace mění parametry soustavy (procesu) a regulátor je nucen přizpůsobit se těmto změnám (adaptace). Aby se regulátor mohl přizpůsobit, je nutné znát chování soustavy reprezentované modelem procesu. Cílem je získat model, který co nejlépe aproximuje reálný proces.

Při tvorbě modelu je snahou najít funkci f, která popisuje chování výstupu soustavy $y(t)$ jako funkci vstupních veličin (např. akční veličina $u(t)$, měřitelná poruchová veličina $v(t)$ apod.) [1], [2]:

$$y(t) = f(u(t), v(t), t) \tag{2.0.1}$$

Reálný proces zahrnuje i náhodné (stochastické) vlivy (např. změna surovin, neměřitelná porucha apod.) [1], [2]:

$$y(t) = f(u(t), v(t), t) + n(t)\ , \tag{2.0.2}$$

kde $n(t)$ reprezentuje stochastické vlivy

2.1 NÁVRH MODELU

Při návrhu adaptivních regulátorů se vychází nejčastěji z regresního (ARX) modelu soustavy [1], [2]. Výstup soustavy je:

$$y(k) = -\sum_{i=1}^{na} a_i y(k-i) + \sum_{i=1}^{nb} b_i u(k-i) + \sum_{i=1}^{nd} d_i v(k-i) + e_s(k) \tag{2.1.1}$$

kde

v měřitelný šum

e_s náhodný šum (neměřitelný)

Jiný způsob zápisu:

$$A(q^{-1})\, y(k) = B(q^{-1})\, u(k) + D(q^{-1})\, v(k) + e_s(k) \tag{2.1.2}$$

Regresní model ARX se často zapisuje v kompaktní vektorové formě:

$$y(k) = \Theta^{T}(k)\phi(k-1) + e_s(k) \tag{2.1.3}$$

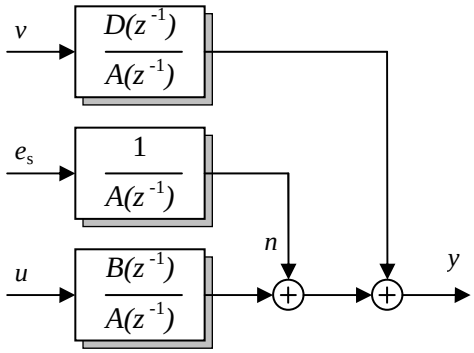
kde

$$\Theta^{T}(k) = [a_1, a_2, ..., a_{na}, b_1, b_2, ..., b_{nb}, d_1, d_2, ..., d_{nd}] \tag{2.1.4}$$

je vektor parametrů vyšetřovaného modelu (neznámé parametry) a

$$\begin{aligned}\phi^{T}(k-1) = [&-y(k-1), -y(k-2), ..., -y(k-na), u(k-1),\\ &u(k-2), ..., u(k-nb), v(k-1), v(k-2), ..., v(k-nd)]\end{aligned} \tag{2.1.5}$$

je vektor dat, tzv. regresor.



Obrázek 2.1: Blokové schéma regresního modelu ARX [1]

2.1.1 Postup získání parametrů modelu

- volba řádu modelu procesu
- volba metody identifikace
- ověření shodnosti chování procesu a modelu

2.1.2 Volba řádu modelu

Nejprve je nutné se rozhodnout, jakým modelem (řádem) se bude daný proces aproximovat. Nejčastější volbou je soustava druhého řádu, soustava druhého řádu s dopravním zpožděním a soustava třetího řádu.

Model druhého řádu lze zapsat jako:

$$F(z)=\frac{Y(z)}{U(z)}=\frac{b_1 z^{-1}+b_2 z^{-2}}{1+a_1 z^{-1}+a_2 z^{-2}} \tag{2.1.6}$$

Model druhého řádu s dopravním zpožděním $T_Z < T_S$ lze zapsat jako:

$$F(z)=\frac{Y(z)}{U(z)}=\frac{b_1 z^{-1}+b_2 z^{-2}+b_3 z^{-3}}{1+a_1 z^{-1}+a_2 z^{-2}} \tag{2.1.7}$$

Model druhého řádu s dopravním zpožděním $T_Z = n \cdot T_S$ lze zapsat jako:

$$F(z)=\frac{Y(z)}{U(z)}=\frac{\left(b_1 z^{-1}+b_2 z^{-2}\right)\cdot z^{-n}}{1+a_1 z^{-1}+a_2 z^{-2}} \tag{2.1.8}$$

Model druhého řádu s dopravním zpožděním $T_Z = n \cdot T_S + \Delta$ lze zapsat jako:

$$F(z)=\frac{Y(z)}{U(z)}=\frac{\left(b_1 z^{-1}+b_2 z^{-2}+b_3 z^{-3}\right)\cdot z^{-n}}{1+a_1 z^{-1}+a_2 z^{-2}} \tag{2.1.9}$$

Model třetího řádu lze zapsat jako

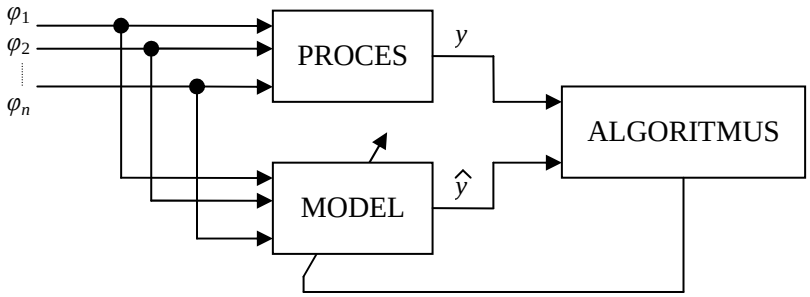
$$F(z)=\frac{Y(z)}{U(z)}=\frac{b_1 z^{-1}+b_2 z^{-2}+b_3 z^{-3}}{1+a_1 z^{-1}+a_2 z^{-2}+a_3 z^{-3}} \tag{2.1.10}$$

2.2 METODA NEJMENŠÍCH ČTVERCŮ

(Method of Least Squares)

Jedná se o matematickou metodu, pomocí které lze nalézt vhodnou aproximační funkci pro dané hodnoty. Tato metoda minimalizuje kvadrát odchylek výstupních hodnot ze soustavy a výstupních hodnot z modelu. Výstupem metody nejmenších čtverců jsou koeficienty hledané funkce.

Metoda je popsána obecně pro *n* vstupů (nebo vnitřních stavů) [2].



Obrázek 2.2: Zobrazení souvislosti mezi procesem a modelem soustavy

Model je popsán experimentální rovnicí:

$$\hat{y}(i) = \varphi_1(i)\theta_1 + \varphi_2(i)\theta_2 + ... + \varphi_n(i)\theta_n + \varepsilon = \varphi^{\mathrm{T}}(i)\theta + \varepsilon$$
$$\theta = (\theta_1 \quad \theta_2 \quad \dots \quad \theta_n)^{\mathrm{T}} \qquad (2.2.1)$$
$$\varphi^{\mathrm{T}}(i) = (\varphi_1(i) \quad \varphi_2(i) \quad \dots \quad \varphi_n(i))$$

kde

$\hat{y}$	odhad výstupní veličiny modelu
θ	hledaný vektor neznámých parametrů
φ	vektor známých měřených funkcí
i	krok výpočtu
ε	chyba v kroku výpočtu

Proměnné φ_i se označují jako regresní proměnné, model je proto nazýván regresním modelem. Po n měřeních lze určit první odhad parametrů modelu (předpoklad: parametry vektoru θ se nemění).

Generování výstupní veličiny lze maticově vyjádřit rovnicí:

$$y = \Phi\theta + \varepsilon \qquad (2.2.2)$$

Rovnici (2.2.2) lze rozepsat:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1n} \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_{n1} & \varphi_{n2} & \cdots & \varphi_{nn} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix} \qquad (2.2.3)$$

Hledá se minimum účelové funkce:

$$J(\theta)=\frac{1}{2}\varepsilon^{\mathrm{T}}\varepsilon=\frac{1}{2}\|\varepsilon\|^{2}=\frac{1}{2}(y-\Phi\theta)^{\mathrm{T}}(y-\Phi\theta) \qquad (2.2.4)$$

Minimum získáme, pokud derivaci podle θ položíme rovnu 0:

$$\left.\frac{\partial J}{\partial\theta}\right|_{\theta=\hat{\theta}}=0 \qquad (2.2.5)$$

Řešením je:

$$\theta=\left(\Phi^{\mathrm{T}}\Phi\right)^{-1}\Phi^{\mathrm{T}}y \qquad (2.2.6)$$

kde

$$P(i)=\left(\Phi^{\mathrm{T}}\Phi\right)^{-1} \qquad \text{je kovarianční matice} \qquad (2.2.7)$$

Nevýhoda této metody: matice se vždy rozšíří o 1 řádek v každém kroku. Z toho plyne velká náročnost na paměť a výpočetní dobu vyhodnocovacího zařízení.

2.2.1 Průběžná metoda nejmenších čtverců

(RLS – Recursive Least Squares)

Při klasické metodě nejmenších čtverců by se matice Φ v každém kroku postupně rozšířila o 1 řádek. V průběžné metodě nejmenších čtverců je do stávajícího rozměru začleněno další měření [2].

Platí:

$$\theta(i)=P(i)\Phi^{\mathrm{T}}(i)y \qquad (2.2.8)$$

v kroku $i+1$ se rozšíří matice o další řádek

$$\begin{bmatrix} y_i \\ y_{i+1} \end{bmatrix}=\begin{bmatrix} \varphi_{i,1} & \varphi_{i,2} & \cdots & \varphi_{i,n} \\ \varphi_{i+1,1} & \varphi_{i+2,1} & \cdots & \varphi_{i+1,n} \end{bmatrix}\begin{bmatrix} \theta_i \\ \theta_{i+1} \end{bmatrix}+\begin{bmatrix} \varepsilon_i \\ \varepsilon_{i+1} \end{bmatrix} \qquad (2.2.9)$$

$$\theta(i+1)=P(i+1)\left[\Phi^{\mathrm{T}}(i),\varphi(i+1)\right]\begin{bmatrix} y_i \\ y_{i+1} \end{bmatrix} \qquad (2.2.10)$$

Pozn.: Podrobné upravení je popsáno v [2], str. 65-67.

Výsledné rovnice po zapsání do rekurentních vztahů jsou:

$$\begin{aligned}\theta(i+1) &= \theta(i) + K(i+1)\left[y(i+1) - \varphi^{\mathrm{T}}(i+1)\theta(i)\right] \\ K(i+1) &= P(i)\varphi(i+1)\left[1 + \varphi^{\mathrm{T}}(i+1)P(i)\varphi(i+1)\right]^{-1} \\ P(i+1) &= P(i) - K(i+1)\varphi^{\mathrm{T}}(i+1)P(i)\end{aligned} \qquad (2.2.11)$$

kde člen $\theta(i+1)$ určuje odhad parametrů pro následující krok, který se počítá z odhadu v kroku předcházejícím - tento krok je opraven o hodnotu úměrnou rovnici

$$y(i+1) - \varphi^{\mathrm{T}}(i+1)\theta(i) \qquad (2.2.12)$$

a člen $K(i+1)$ je váhový součinitel - určuje, s jakým vlivem se má tento rozdíl vzít v úvahu a tím urychlit (zpozdit) nový výpočet parametrů.

Výhoda průběžné metody: matice se nerozšiřuje, vždy se nahradí stávající hodnoty novým měřením.

2.2.2 Exponenciální zapomínání

Pokud má algoritmus sledovat i pomalé změny parametrů identifikovaného procesu, lze použít tzv. exponenciální zapomínání [2], které minimalizuje modifikované kritérium:

$$J_k(\theta) = \sum_{i=k_0}^{k} \lambda^{2(k-1)} \varepsilon^2(i) \qquad (2.2.13)$$

kde

$0 < \lambda^2 \leq 1$ je faktor exponenciálního zapomínání.

Po úpravě lze získat modifikované vztahy:

$$\begin{aligned}\theta(i+1) &= \theta(i) + K(i+1)\left[y(i+1) - \varphi^{\mathrm{T}}(i+1)\theta(i)\right] \\ K(i+1) &= P(i)\varphi(i+1)\left[\lambda_{i+1} + \varphi^{\mathrm{T}}(i+1)P(i)\varphi(i+1)\right]^{-1} \\ P(i+1) &= \frac{1}{\lambda_{i+1}}\left[P(i) - K(i+1)\varphi^{\mathrm{T}}(i+1)P(i)\right]\end{aligned} \qquad (2.2.14)$$

Jako počáteční nastavení algoritmu se nejčastěji volí:

$$P_0 = 10^4 - 10^6 \; I$$
$$\theta_0 = [1 \quad 0 \quad 0 \quad \ldots \quad 0]^{\mathrm{T}} \tag{2.2.15}$$
$$\lambda_0 = 0{,}995 - 0{,}999$$

kde

I - jednotková matice

Např. pro model druhého řádu s dopravním zpožděním platí:

$$F(z) = \frac{Y(z)}{U(z)} = \frac{b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{2.2.16}$$

$$\varphi^{\mathrm{T}} = (u(i-1) \quad u(i-2) \quad u(i-3) \quad -y(i-1) \quad -y(i-2)) \tag{2.2.17}$$

Potom je odhad parametrů:

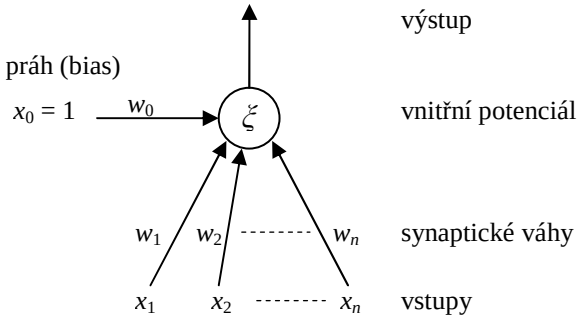
$$\theta = \left(\hat{b}_1 \quad \hat{b}_2 \quad \hat{b}_3 \quad \hat{a}_1 \quad \hat{a}_2\right)^{\mathrm{T}} \tag{2.2.18}$$

2.3 NEURONOVÉ SÍTĚ

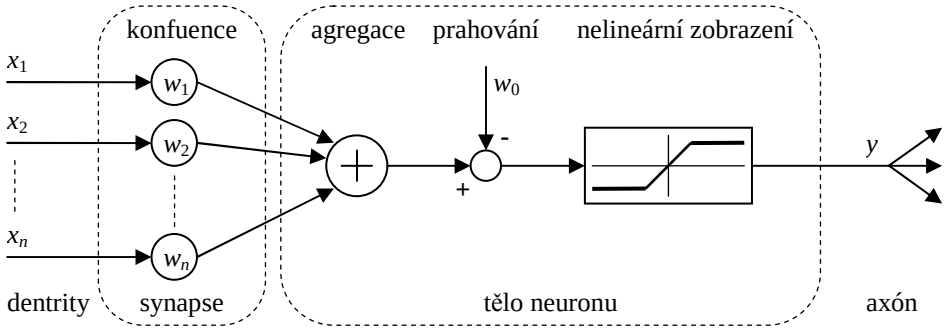
(NN – Neural Network)

Neuron má obecně n reálných vstupů $x_1, x2, ..., x_n$. Vstupy jsou ohodnoceny synaptickými váhami $w_1, w_2, ..., w_n$, které určují jejich propustnost. Suma vstupních hodnot představuje vnitřní potenciál neuronu. Každý neuron má ještě jeden vstup nazvaný práh (bias) s váhou w_0, který je připojen na hodnotu 1. Výstup z neuronu bývá označován jako y (nebo u).

Existuje analogie mezi biologickým a umělým neuronem - **Obrázek 2.4**.



Obrázek 2.3: Obecný model neuronu



Obrázek 2.4: Schéma analogie biologického a umělého neuronu

2.3.1 Agregační a aktivační funkce neuronu

Agregační funkce neuronu x_a má za úkol sloučit určitým způsobem vstupní signály x_i neuronu [6]. Tato funkce transformuje $n+1$ rozměrný vstupní vektor x_i na skalární signály x_a. Obecně lze agregaci zapsat jako:

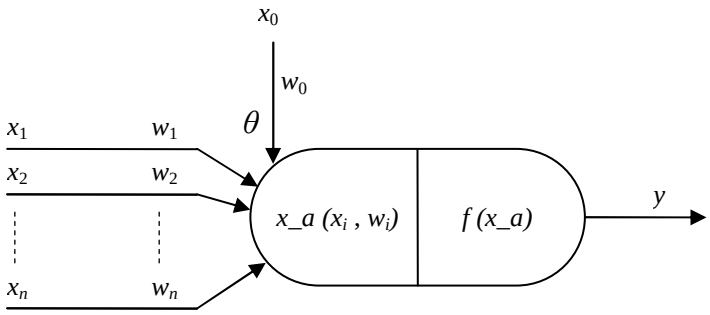
$$x\_a(k) = \sum_{i=1}^{n} x_i(k) \cdot w_i(k) + \theta \qquad (2.3.1)$$

Lze zavést substituci:

$$\theta = x_0 \cdot w_0 \qquad (2.3.2)$$

Potom je výsledný vztah:

$$x\_a(k) = \sum_{i=0}^{n} x_i(k) \cdot w_i(k) \qquad (2.3.3)$$

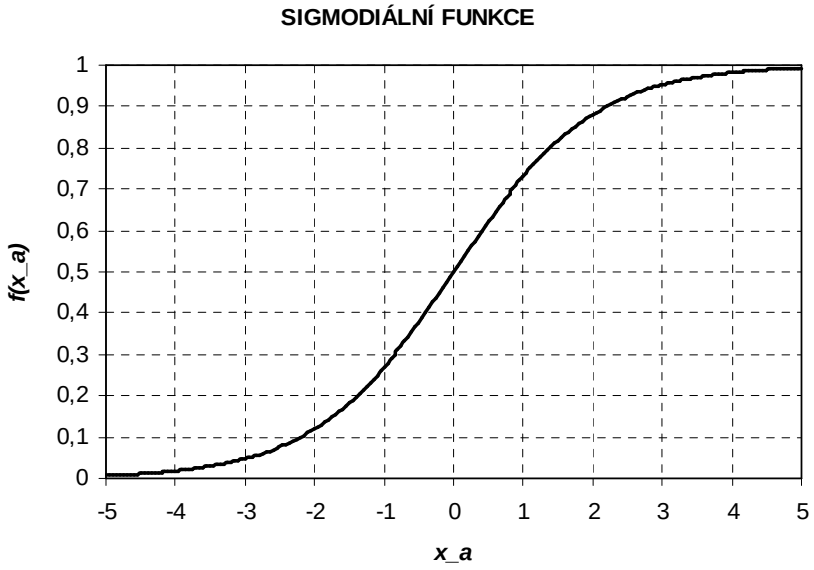


Obrázek 2.5: Vnitřní schéma umělého neuronu

kde x_a - agregační funkce neuronu (vstupní potenciál neuronu)

f - aktivační funkce neuronu

θ - práh neuronu



Obrázek 2.6: Příklad průběhu sigmodiální aktivační funkce neuronu

Aktivační funkce neuronu $f(x\_a)$ převádí hodnotu vstupního potenciálu na výstupní hodnotu z neuronu. Konkrétní tvary přenosových funkcí jsou různé, dělí se na lineární a nelineární, příp. spojité a diskrétní.

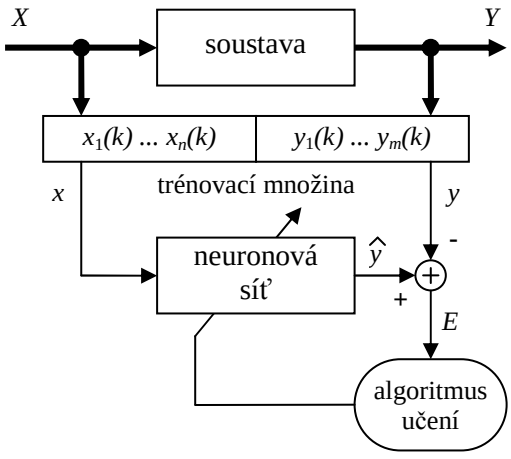
Neuronová síť se skládá z jednotek (neuronů, které mohou obsahovat různé přenosové funkce), propojení mezi nimi a adaptivního algoritmu. Počet neuronů a jejich propojení určuje architektura (topologie) neuronové sítě. Neurony v síti lze podle využití rozdělit na vstupní, pracovní (skryté, mezilehlé) a výstupní.

Například: Funkční předpis sigmodiální aktivační funkce (**Obrázek 2.6**):

$$f(y\_a) = \frac{1}{1+e^{-y\_a}} \tag{2.3.4}$$

2.3.2 Trénování neuronové sítě

Proces učení probíhá tak, že na vstupy sítě se opakovaně připojuje soubor vzorů požadovaného chování sítě, tzv. trénovací množina (TRM) a hledá se optimální nastavení parametrů sítě (vektor vah), při kterém bude chyba E minimální (rozdíl mezi výstupem modelované soustavy a neuronové sítě). Trénovací množina se získá experimentálně měřením příslušných vstupních a výstupních veličin modelovaného procesu nebo z matematického popisu procesu při numerické simulaci. Jeden průchod trénovací množiny neuronovou sítí se nazývá epocha [7].



Obrázek 2.7: Základní schéma trénování NS

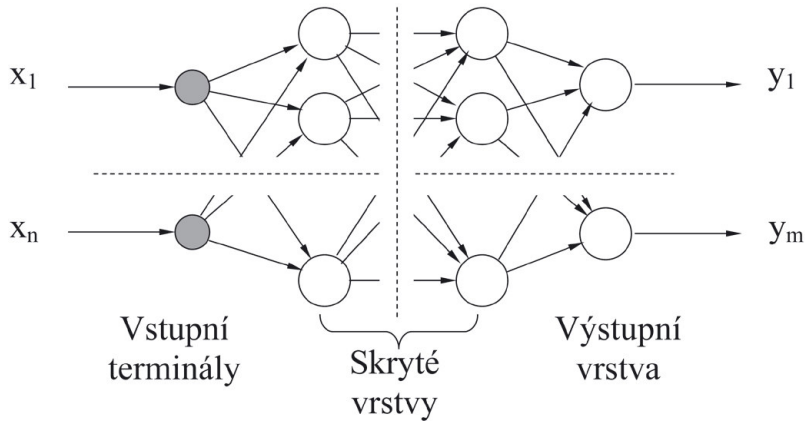
2.3.3 Režimy dynamiky neuronové sítě

Neuronová síť se v čase vyvíjí – váhy se adaptují. Rozdělujeme různé druhy režimů dynamiky [5]:

- organizační (změna topologie) - specifikuje strukturu neuronové sítě, organizační dynamika převážně předpokládá pevnou architekturu, která se již nemění.
- aktivní (změna stavu) - specifikuje počáteční stav sítě a způsob jeho změny v čase při pevné topologii a konfiguraci, v aktivním režimu se na začátku nastaví stavy vstupních neuronů a zbylé neurony jsou v uvedeném počátečním stavu, po inicializaci stavu sítě probíhá vlastní výpočet
- adaptivní (změna konfigurace) - specifikuje počáteční konfiguraci sítě a jakým způsobem se mění váhy v síti v čase, na začátku se nastaví váhy všech spojů v síti na počáteční konfiguraci (např. náhodně), po inicializaci probíhá vlastní adaptace

2.3.4 Dopředná neuronová síť

Pro identifikaci se nejčastěji používá dopředná neuronová síť (Feed Forward Neural Network) označována jako DNS. Tato třída neuronových sítí se od ostatních odlišuje tím, že neurony jsou rozděleny do vrstev a neexistují zde zpětné vazby. Předpokladem je, že neurony dvou sousedních vrstev jsou propojeny každý s každým.



Obrázek 2.8: Vícevrstvá dopředná neuronová síť [7]

Problémem je stanovení optimální struktury – počet vrstev a počet neuronů v jednotlivých vrstvách. Za optimální počet neuronů (vrstev) považujeme minimální počet neuronů (vrstev), se kterým je neuronová síť ještě schopna dostatečně řešit danou úlohu [4]. Ze zkušeností vyplývá, že v řadě případů dosáhneme vyhovující aproximace chování modelované soustavy za použití dvou skrytých vrstev neuronové sítě.

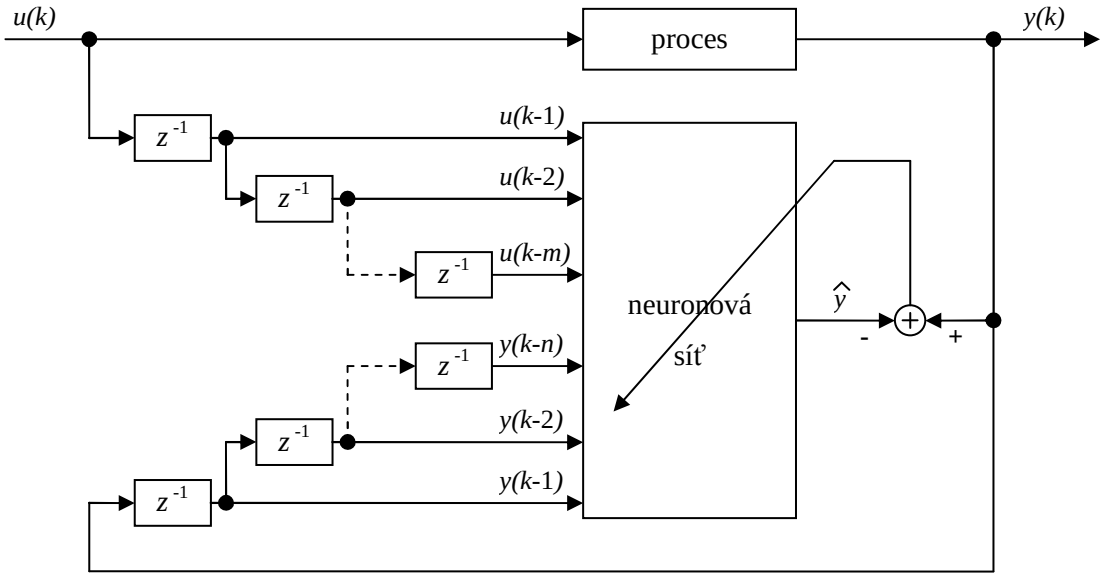
Pozn.: Větší počet skrytých vrstev a velký počet neuronů ve skrytých vrstvách může vést k tzv. „přeučení“ sítě a nevede k lepším výsledkům.

2.3.5 Off-line a on-line učení

Off-line učení – učení neuronového modelu před jeho použitím, učí se na základě historických dat bez připojení k systému. Výhody - neomezenost časem, ověření jednotlivých variant zapojení neuronových modelů, můžeme měnit např. i topologii sítě.

On-line učení – učení neuronového modelu během chodu systému, v každém kroku se modifikuje nastavení podle současného stavu, používá se zapomínání [1].

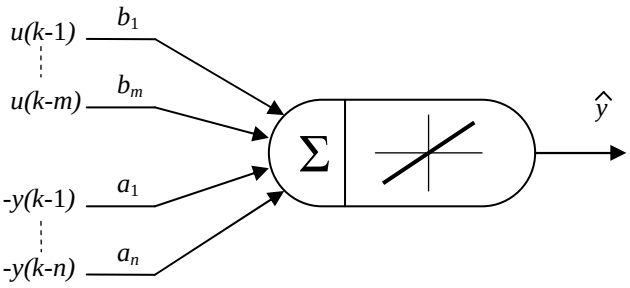
2.3.6 Identifikace soustavy pomocí neuronové sítě



Obrázek 2.9: Schéma identifikace založené na neuronové síti

Na **Obrázku 2.9** je znázorněna identifikace (tj. učení) neuronové sítě a porovnávání se skutečnou soustavou. Výhodné je použít neuronovou síť s 1 neuronem, který má lineární aktivační funkci

$$f(a\_x) = a\_x \tag{2.3.5}$$



Obrázek 2.10: Nejjednodušší struktura pro identifikaci s 1 neuronem

Potom lze jednotlivé vstupní váhy považovat přímo za konstanty přenosu soustavy v diskrétním tvaru

$$F(z) = \frac{Y(z)}{U(z)} = \frac{b_1 z^{-1} + b_2 z^{-2} + \ldots + b_m z^{-m}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}} \tag{2.3.6}$$

Výstupní predikce je dána:

$$\hat{y} = \sum_{i=1}^{m} b_i \cdot u(k-i) - \sum_{j=1}^{n} a_j \cdot y(k-j) \tag{2.3.7}$$

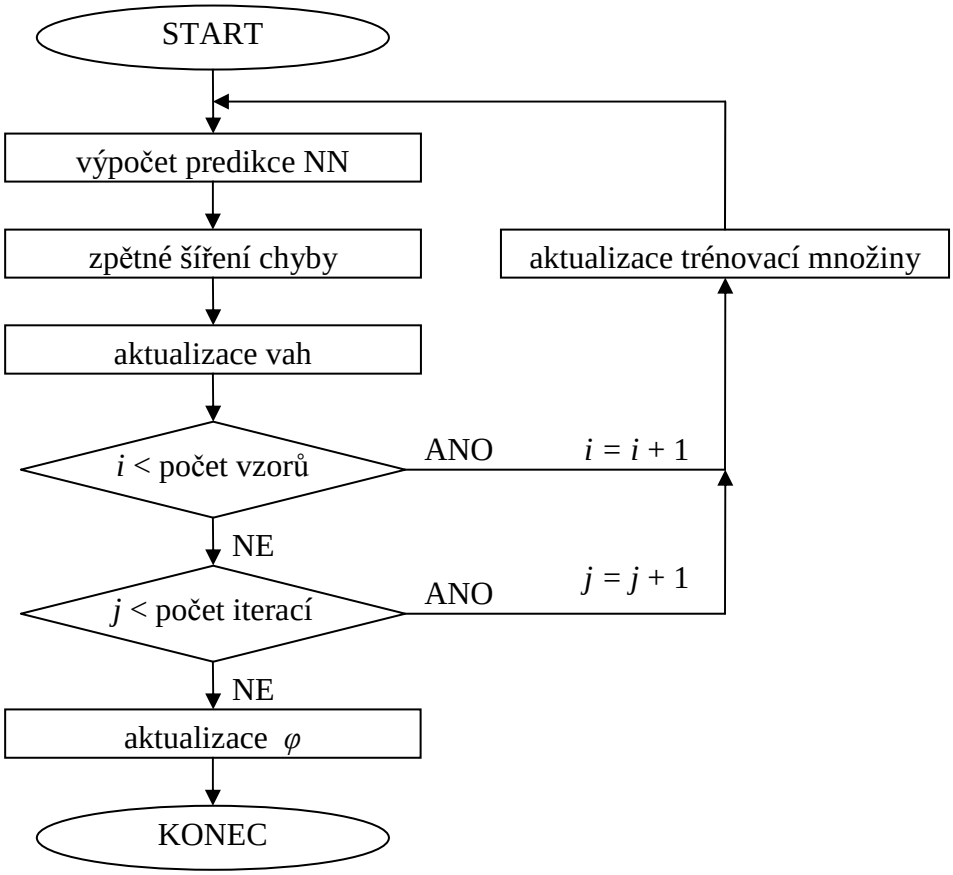
kde m počet členů v čitateli

n počet členů ve jmenovateli

2.3.6.1 Metoda Back-propagation

Pro identifikaci soustavy v řízení (nalezení přenosové funkce) se často používá metoda Back-propagation (zpětné šíření chyby). Snahou je minimalizovat funkci:

$$V = \frac{1}{2} e^{\mathrm{T}} e \tag{2.3.8}$$



Obrázek 2.11: Algoritmus učení neuronové sítě metodou Back-propagation

Algoritmus lze obecně zapsat:

- Výpočet výstupu (predikce) NN s lineární aktivační funkcí

$$\hat{y} = f\left(\sum_i \varphi(i)\cdot w(i)\right) \quad \Rightarrow \quad \hat{y} = \varphi^{\mathrm{T}}\cdot w \tag{2.3.9}$$

kde

$$\varphi = \left[u(k-1) \quad \ldots \quad u(k-m) \quad -y(k-1) \quad \ldots \quad -y(k-n)\right]^{\mathrm{T}} \tag{2.3.10}$$

$$w = \left[w_1 \quad \ldots \quad w_m \quad w_{m+1} \quad \ldots \quad w_{m+n}\right]^{\mathrm{T}} \tag{2.3.11}$$

zároveň lze uvažovat, že

$$w \approx \left[b_1 \quad \ldots \quad b_m \quad a_1 \quad \ldots \quad a_n\right]^{\mathrm{T}} \tag{2.3.12}$$

- Zpětné šíření chyby

$$\delta = e = y - \hat{y} \tag{2.3.13}$$

- Aktualizace vah

$$w(k+1) = w(k) + \eta \cdot (w(k) - w(k-1)) + \mu \cdot \delta \cdot \varphi \qquad (2.3.14)$$

kde

η - setrvačnost (momentum)

μ - konstanta učení (learning rate), má vliv na rychlost konvergence

2.3.6.2 Metoda Levenberg-Marquardt

Metoda Levenberg-Marquardt je iterační metoda, která řeší problém minimalizace sumy kvadrátu odchylek obecné nelineární funkce [16]. Princip metody je založený na hledání globálního minima chyby minulých výstupů z modelované soustavy a výstupů z modelu přes paměť posledních hodnot (trénovací množiny):

$$X(k) = [\varphi(k) \quad \varphi(k-1) \quad \ldots \quad \varphi(k-p)] \qquad (2.3.15)$$

Trénovací množina zahrnuje určitý počet posledních stavů systému $[\varphi(k) \quad \ldots \quad \varphi(k-p)]$, kde p je velikost bufferu. Algoritmus pro určitý počet iterací i v každém kroku identifikace k lze zapsat následovně:

$$\theta(i \mid k+1) = \theta(i \mid k) - \left[J^{\mathrm{T}}(i \mid k)\, J^{\mathrm{T}}(i \mid k) + \mu I\right]^{-1} J^{\mathrm{T}}(i \mid k) E(i \mid k) \qquad (2.3.16)$$

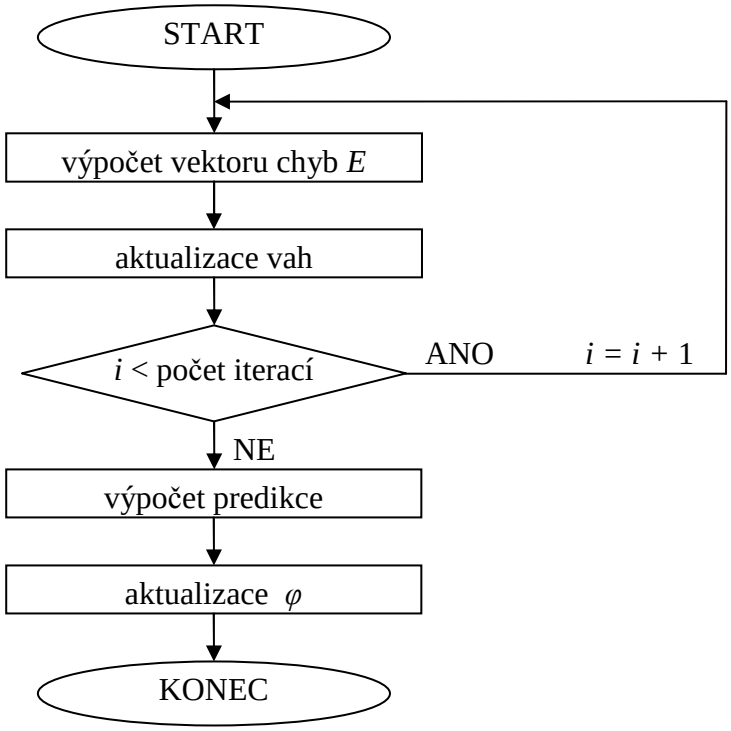
kde μ je krok učení, I je jednotková matice, $E(i \mid k)$ je vektor chyb mezi výstupem modelovaného systému $T(k)$ a výstupem modelu:

$$E(k) = T^{\mathrm{T}}(k) - X^{\mathrm{T}}(k)\theta(k) \qquad (2.3.17)$$

$$T(k) = [y(k) \quad y(k-1) \quad \ldots \quad y(k-p)] \qquad (2.3.18)$$

Matice $J(i \mid k)$ reprezentuje nejlepší lineární aproximaci k diferencovatelné funkci (tj. vektoru hodnot) blízko danému bodu:

$$J(k) = \frac{\partial E(k)}{\partial \theta(k)} = \frac{\partial \left(T^{\mathrm{T}}(k) - X^{\mathrm{T}}(k)\theta(k)\right)}{\partial \theta(k)} = -X^{\mathrm{T}}(k) \qquad (2.3.19)$$



Obrázek 2.12: Algoritmus učení neuronové sítě metodou Levenberg-Marquardt

2.4 VYHODNOCENÍ IDENTIFIKACE

Zvolené soustavy:

$$F_1(s) = \frac{0{,}5}{(s+1)^2} = \frac{0{,}5}{s^2 + 2s + 1} \tag{2.4.1}$$

$$F_2(s) = \frac{0{,}5}{s(s+1)^2} = \frac{0{,}5}{s^3 + 2s^2 + s} \tag{2.4.2}$$

Perioda vzorkování byla pro všechny metody vybrána stejně $T_S = 0{,}5\,\mathrm{s}$. Jednalo se vždy o model 3. řádu a doba simulace probíhala po dobu 150s. Metoda RLS byla nastavena s exponenciálním zapomínáním $\lambda = 0{,}995$, metoda BP s velikostí trénovací množiny $p = 50$, konstantou učení $\mu = 0{,}01$ a momentem $\eta = 0{,}5$. Metoda LM byla nastavena se stejnou velikostí trénovací množinou $p = 50$ a konstantou učení $\mu = 0{,}01$.

Tyto metody byly porovnávány pomocí lineárního (2.4.3), kvadratického (2.4.4) a ITAE (2.4.5) kritéria:

$$J_{\mathrm{LIN}} = \sum_{i=0}^{k} \left| y - \hat{y} \right| \tag{2.4.3}$$

$$J_{\mathrm{KV}} = \sum_{i=0}^{k} \left(y - \hat{y} \right)^2 \tag{2.4.4}$$

$$J_{\mathrm{ITAE}} = \sum_{i=0}^{k} k \left| y - \hat{y} \right| \tag{2.4.5}$$

soustava	metoda	lineární krit.	kvadrat. krit.	ITAE krit.
$F_1(s)$	RLS	1,2782	1,0183	2,6891
	BP	3,0992	1,4613	16,2320
	LM	0,0144	0,0002	0,1569
$F_2(s)$	RLS	1,3620	1,0134	7,7280
	BP	selhává	selhává	selhává
	LM	0,0582	0,0004	0,6011

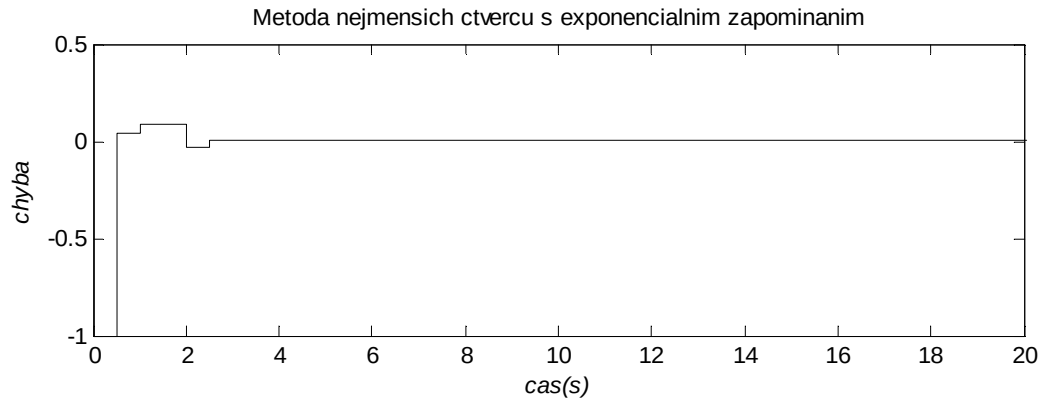
Tabulka 2.1: Výsledné hodnoty kritérií

U metody nejmenších čtverců dochází k největší chybě vždy v 1. kroku identifikace. Je to způsobené náhodnou volbou prvotního nastavení koeficientů odhadu Θ. Dále bylo ověřeno, že při nastavení $\lambda = 1$ (tj. bez exponenciálního zapomínání) se model naučí již po 6. kroku (pokud se jedná konkrétně o model 3. řádu).

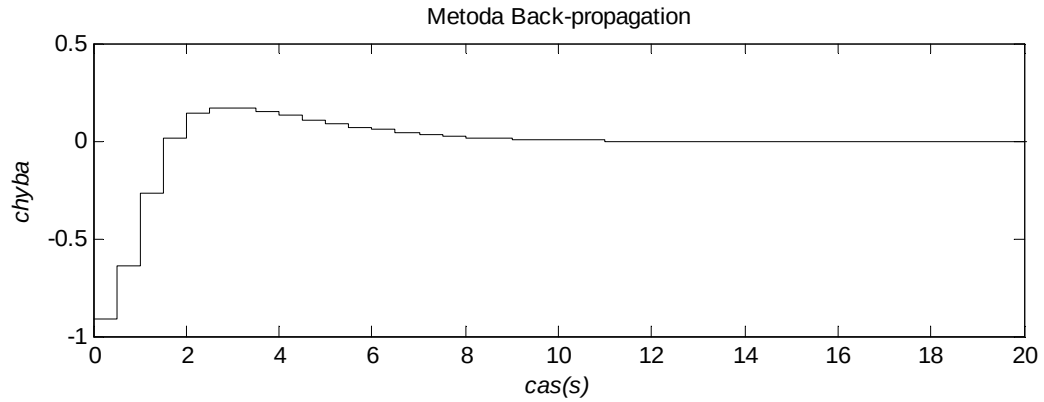
Z pohledu výpočetní náročnosti je nejpříznivější metoda RLS. Tato metoda je iterační (neklade velké nároky na paměť) a neprovádí výpočet inverzních matic. Metoda BP vyžaduje větší paměť, ale na rozdíl od metody LM se zde opět neprovádí výpočet inverzních matic.

Z **Tabulky 2.1** vyplývá, že takto nastavená metoda LM vycházela příznivěji než jiné metody. Proto byla použita pro praktické ověřování na fyzikálním modelu.

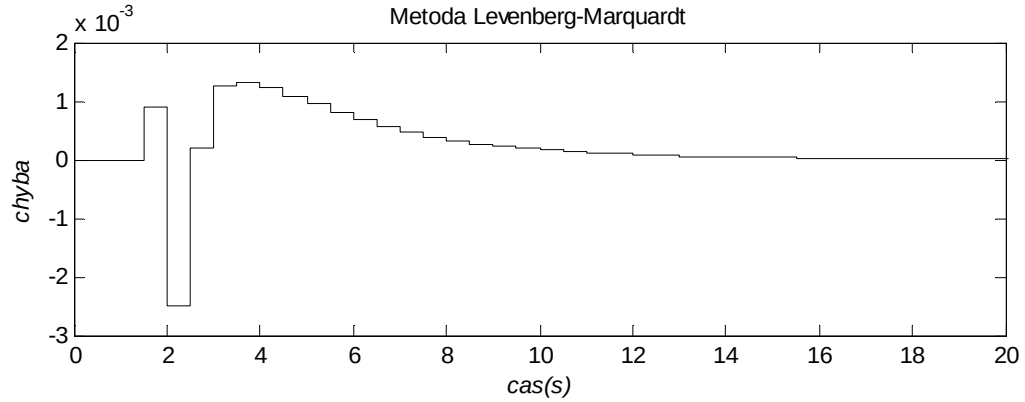
Algoritmy pro výpočetní program MATLAB jednotlivých metod jsou uvedeny v **Příloze 1, 2, 3**.



Obrázek 2.13: Průběh chyby predikce modelu metodou nejmenších čtverců



Obrázek 2.14: Průběh chyby predikce modelu metodou Back-propagation



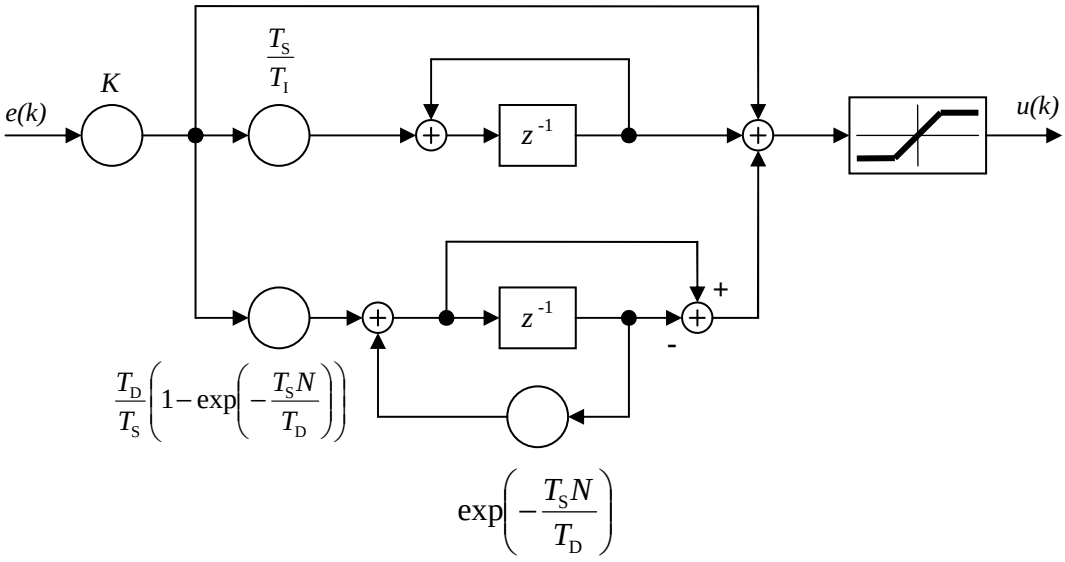
Obrázek 2.15: Průběh chyby predikce modelu metodou Levenberg-Marquardt

3. REGULÁTORY

3.1 DISKRÉTNÍ EKVIVALENTY SPOJITÝCH PID

3.1.1 PSD regulátor s filtrací derivační složky

Zkracováním periody vzorkování roste velikost amplitudy derivační složky. Proto se velmi často používá upravená filtrace derivační složky.



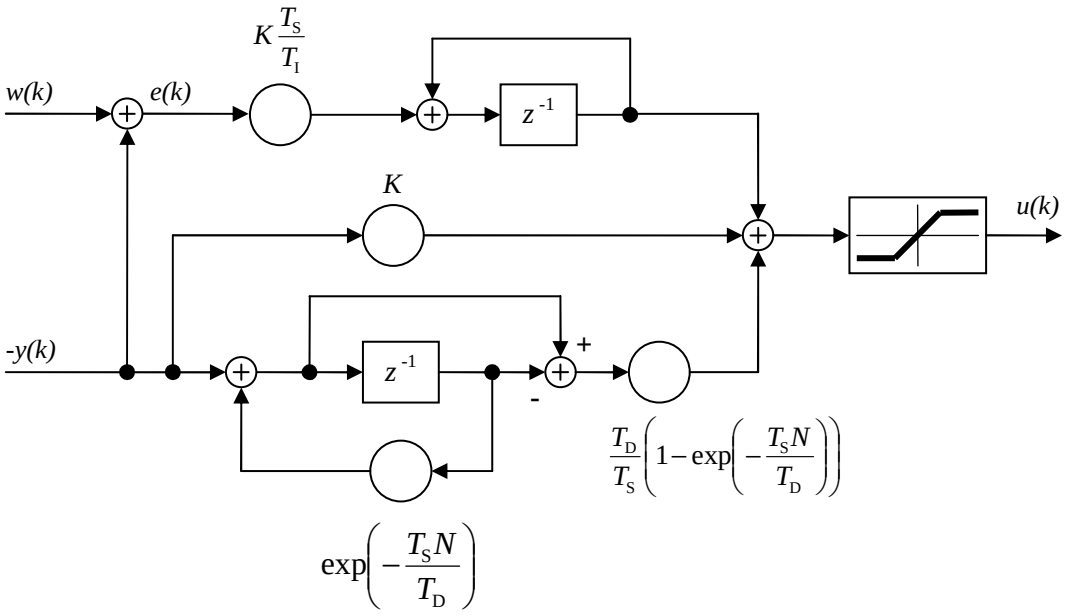
Obrázek 3.1: PSD regulátor s filtrací derivační složky

kde

$e(k)$	odchylka výstupu systému od žádané hodnoty
$u(k)$	akční zásah
T_S	perioda vzorkování
K	proporcionální zesílení
T_I	integrační časová konstanta
T_D	derivační časová konstanta
N	filtrační konstanta, volba 3÷20

3.1.2 S-PD regulátor

Upraveným zapojením lze do jisté míry zmenšit velikost prvního překmitu přechodného děje. Jediný rozdíl oproti klasickému PSD regulátoru je takový, že do proporcionální a diferenční složky regulátoru přivádíme záporný signál z řízené soustavy.



Obrázek 3.2: S-PD regulátor s filtrací derivační složky

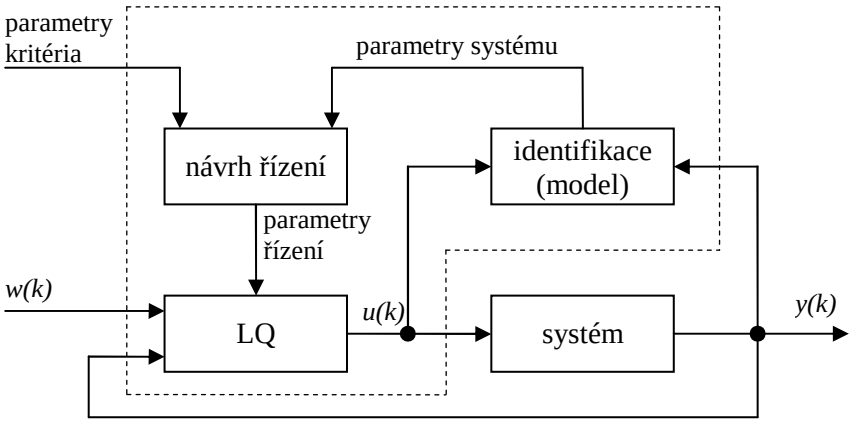
kde

$w(k)$	žádaná hodnota
$e(k)$	odchylka výstupu systému od žádané hodnoty
$u(k)$	akční zásah
$y(k)$	výstup systému
T_S	perioda vzorkování
K	proporcionální zesílení
T_I	integrační časová konstanta
T_D	derivační časová konstanta
N	filtrační konstanta, volba 3÷20

Uvedené regulátory byly v práci použity pro porovnání řízení (přímo modelovány v programu MATLAB/Simulink). Nastavení jednotlivých parametrů bylo prováděno na základě inženýrského přístupu a doladění provedeno metodou pokus-omyl.

3.2 ADAPTIVNÍ OPTIMÁLNÍ REGULÁTOR

Adaptivní optimální regulátor je typ samočinně se nastavujícího regulátoru, který mění svoje parametry na základě průběžných informací o soustavě s ohledem na kritérium optimality. Při použití identifikace modelu ARX lze identifikované parametry použít přímo ve stavovém modelu regulovaného systému. Nemusí se tedy navrhovat konstanty regulátoru (jako např. u adaptivních PSD regulátorů).



Obrázek 3.3: Blokové schéma adaptivního LQ řízení

3.2.1 Optimální řízení

Pojem „optimální“ - John a James Bernoulli (1696) - problém o brachistrochroně (křivka spojující dva body, po které se má ideální těleso dostat z jednoho bodu do druhého za nejkratší čas).

Daný optimalizační problém lze řešit různými metodami:

- Lagrangeovými multiplikátory
- Bellmanovým dynamickým programováním (1957)
- Pontrjaginovým principem minima (1962)
- Hamiltonovým principem
- Euler-Lagrangeovým principem

Aby se teorie optimálního řízení dala použít v reálném prostředí, je nutné vzít v úvahu [10]:

- numerická konvergence ke správnému řešení (globálnímu extrému)
- hledání vhodného kritéria z pohledu uživatele
- tvrdé omezující podmínky na regulaci (konečný akční zásah)
- časová náročnost výpočtu oproti jiným metodám atd.

3.2.2 Algoritmus LQ řízení

Kvadratický optimální regulátor je ve své základní formě stavový regulátor s proporcionálními zpětnými vazbami od stavů systému [13]. Matice zesílení K se volí na základě optimálního kritéria.

Lineární kvadratický regulátor bývá obecně označovaný jako LQ regulátor (Linear system, Quadratic cost) – jedná se o lineární regulátor s kvadratickým kritériem. Někdy bývá označován jako LQG – Gaussův bílý šum působící na soustavu. Vlastnosti LQ regulátoru se nastavují pomocí konstant optimalizovaného kritéria.

Úkolem optimálního řízení je najít takové řízení $u(k)$, které [10]:

- převádí systém z počátečního stavu do koncového stavu
- patří do třídy přípustných řízení
- minimalizuje dané kritérium

Základní tvar kvadratického kritéria optimálního řízení:

$$J = x^{\mathrm{T}}(N)Sx(N) + \sum_{k=0}^{N-1}\left[x^{\mathrm{T}}(k)Qx(k) + u^{\mathrm{T}}(k)Ru(k)\right] \qquad (3.2.1)$$

kde S ... čtvercová matice, penalizace konečného stavu systému

Q ... čtvercová matice, penalizace odchylek stavů systému od nulových hodnot

R ... čtvercová matice, penalizace energie pro řízení

Pozn.: Matice S a Q jsou pozitivně semidefinitní (skalár $x^{\mathrm{T}}(N)Sx(N)$, resp. $x^{\mathrm{T}}(k)Qx(k)$ musí být nezáporný pro všechna x). Matice R je pozitivně definitní (skalár $u^{\mathrm{T}}(k)Ru(k)$ musí být kladný pro všechna u).

Zavedením nekonečného horizontu (tj. $S = Q$) se kritérium (3.2.1) výrazně zjednoduší (je uvažováno, že řízení probíhá pořád dále - není proto nutná definice konečného stavu):

$$J = \sum_{k=0}^{N}\left[x^{\mathrm{T}}(k)Qx(k) + u^{\mathrm{T}}(k)Ru(k)\right] \qquad (3.2.2)$$

Systém lze popsat:

$$\begin{aligned} x(k+1) &= A\,x(k) + B\,u(k) \\ y(k) &= C\,x(k) \end{aligned} \qquad (3.2.3)$$

Matici zpětným vazeb K lze získat různými způsoby – např. dynamickým programováním. Matice K konverguje k ustálené hodnotě při dostatečném počtu iterací (horizont optimalizace). Pak platí vztah nazývaný *algebraická Riccatiho rovnice*. Řešení lineárního řízení je ve tvaru:

$$u(k) = -K\,x(k), \qquad (3.2.4)$$

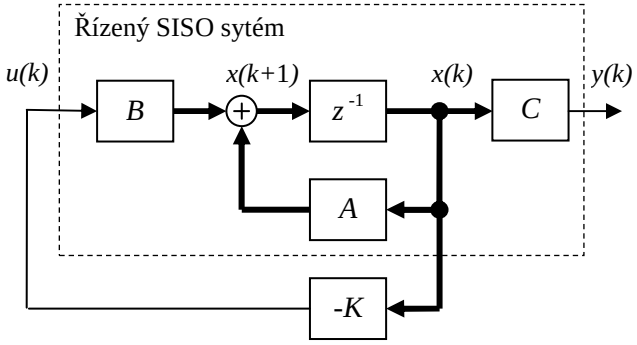
kde matice K je definovaná jako:

$$K = \left[R + B^{\mathrm{T}}PB\right]^{-1}B^{\mathrm{T}}PA \qquad (3.2.5)$$

a matice P je řešením algebraické Riccatiho rovnice:

$$P = Q + K^{\mathrm{T}} RK + (A - BK)^{\mathrm{T}} P(A - BK) \qquad (3.2.6)$$

Pokud je LQ regulátor popsaný tímto způsobem, jedná se o klasický stavový regulátor, který se snaží dostat všechny stavy z počátečních hodnot do nuly. V praxi přibývá požadavek dosažení žádané výstupní hodnoty systému (tzv. sledování referenční trajektorie) a vyregulování působících poruchových veličin bez chyby. Dalším požadavkem bývá dosažení nulové ustálené odchylky (tzv. asymptotické sledování referenční trajektorie). Požadovaných úprav se dosahuje rozšířením struktury modelu. Parametry řídicího algoritmu jsou pak navrženy na rozšířený model.



Obrázek 3.4: Blokové schéma LQ regulátoru v základním tvaru

3.2.2.1 Sledování referenční trajektorie

Úloha kvadraticky optimálního sledování referenční trajektorie je definována tak, že referenční trajektorie je generována systémem jako odezva na jeho počáteční podmínky. Pokud se přivede na vstup tohoto systému signál, dojde ke změně jeho počátečních podmínek pro další část odezvy. Jinými slovy, do regulátoru se zakomponuje matematický model systému generujícího referenční trajektorii [13].

Generátor lze definovat následovně:

$$\begin{aligned} x_{\mathrm{ref}}(k+1) &= x_{\mathrm{ref}}(k) \\ w(k) &= x_{\mathrm{ref}}(k) \end{aligned} \qquad (3.2.7)$$

kde x_{ref} je referenční trajektorie a w je žádaná hodnota.

Dále pro regulační odchylku platí:

$$e(k) = w(k) - y(k) = x_{\text{ref}}(k) - Cx(k) \tag{3.2.8}$$

Nový rozšířený systém (řízený systém a generátor reference) [13]:

$$\begin{bmatrix} x(k+1) \\ x_{\text{ref}}(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x(k) \\ x_{\text{ref}}(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(k) \tag{3.2.9}$$

Pokud se definuje nový stavový vektor

$$x'(k) = \begin{bmatrix} x(k) \\ x_{\text{ref}}(k) \end{bmatrix}, \tag{3.2.10}$$

získá se tím jiný zápis rozšířeného systému pomocí rozšířených matic systému:

$$x'(k+1) = A'x'(k) + B'u(k) \tag{3.2.11}$$

Pokud se výše uvedené vztahy dosadí do kvadratického kritéria, získá se následující zápis:

$$\begin{aligned} J &= \sum_{k=0}^{N} \left[e^{\mathrm{T}}(k) Q\, e(k) + u^{\mathrm{T}}(k) R\, u(k) \right] = \\ &= \sum_{k=0}^{N} \left[(x_{\text{ref}}(k) - Cx(k))^{\mathrm{T}} Q (x_{\text{ref}}(k) - Cx(k)) + u^{\mathrm{T}}(k) R\, u(k) \right] = \\ &= \sum_{k=0}^{N} \left[\begin{bmatrix} x(k) & x_{\text{ref}}(k) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} -C^{\mathrm{T}} \\ I \end{bmatrix} Q \begin{bmatrix} -C & I \end{bmatrix} \begin{bmatrix} x(k) & x_{\text{ref}}(k) \end{bmatrix} + u^{\mathrm{T}}(k) R\, u(k) \right] = \\ &= \sum_{k=0}^{N} \left[x'^{\mathrm{T}}(k) \begin{bmatrix} C^{\mathrm{T}}QC & -C^{\mathrm{T}}Q \\ -QC & Q \end{bmatrix} x'(k) + u^{\mathrm{T}}(k) R\, u(k) \right] \end{aligned} \tag{3.2.12}$$

kde I je jednotková matice.

Pokud se zvolí

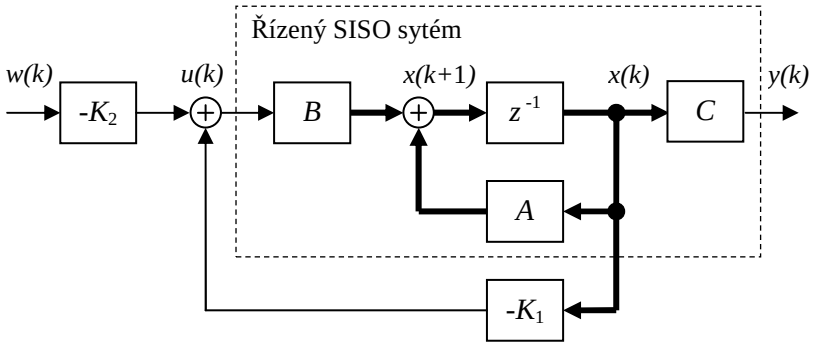
$$Q' = \begin{bmatrix} C^{\mathrm{T}}QC & -C^{\mathrm{T}}Q \\ -QC & Q \end{bmatrix}, \tag{3.2.13}$$

lze řešit Riccatiho rovnici s použitím matic A', B' a Q' místo matic A, B a Q. Výsledkem je matice zesílení K' ve tvaru:

$$K' = \begin{bmatrix} K_1 & K_2 \end{bmatrix} \tag{3.2.14}$$

Optimální akční zásah se určí podle vztahu:

$$u(k) = -K_1 x(k) - K_2 x_{\text{ref}}(k) = -K' x'(k) \tag{3.2.15}$$



Obrázek 3.5: LQ regulátor se sledováním reference

3.2.2.2 Asymptotické sledování referenční trajektorie

Aby bylo možné dosáhnout nulové ustálené odchylky, musí se popis systému rozšířit o sumační člen (integrační akce):

$$\begin{aligned} x_{\text{se}}(k+1) &= x_{\text{se}}(k) + e(k) = x_{\text{se}}(k) + x_{\text{ref}}(k) - Cx(k) \\ y_{\text{se}}(k) &= x_{\text{se}}(k) \end{aligned} \tag{3.2.16}$$

Stavový popis rozšířeného systému se sumátorem má tvar:

$$\begin{bmatrix} x(k+1) \\ x_{\text{ref}}(k+1) \\ x_{\text{se}}(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 & 0 \\ 0 & 1 & 0 \\ -C & 1 & 1 \end{bmatrix} \begin{bmatrix} x(k) \\ x_{\text{ref}}(k) \\ x_{\text{se}}(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix} u(k) = A'' x''(k) + B'' u(k) \tag{3.2.17}$$

Kvadratické kritérium obsahující sumační člen bude ve tvaru:

$$J = \sum_{k=0}^{N} \left[e^{\mathrm{T}}(k) Q\, e(k) + y_{\mathrm{se}}{}^{\mathrm{T}}(k) Q_{\mathrm{se}}\; y_{\mathrm{se}}(k) + u^{\mathrm{T}}(k) R\, u(k) \right] =$$

$$= \sum_{k=0}^{N} \left[\left(x_{\mathrm{ref}}(k) - Cx(k)\right)^{\mathrm{T}} Q \left(x_{\mathrm{ref}}(k) - Cx(k)\right) + x_{\mathrm{se}}{}^{\mathrm{T}}(k) Q_{\mathrm{se}}\; x_{\mathrm{se}}(k) + u^{\mathrm{T}}(k) R\, u(k) \right] =$$

$$= \sum_{k=0}^{N} \left[x'^{\mathrm{T}}(k) \begin{bmatrix} C^{\mathrm{T}}QC & -C^{\mathrm{T}}Q \\ -QC & Q \end{bmatrix} x'(k) + x_{\mathrm{se}}{}^{\mathrm{T}}(k) Q_{\mathrm{se}}\; x_{\mathrm{se}}(k) + u^{\mathrm{T}}(k) R\, u(k) \right] =$$

$$= \sum_{k=0}^{N} \left[\begin{bmatrix} x(k) & x_{\mathrm{ref}}(k) & x_{\mathrm{se}}(k) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} C^{\mathrm{T}}QC & -C^{\mathrm{T}}Q & 0 \\ -QC & Q & 0 \\ 0 & 0 & Q_{\mathrm{se}} \end{bmatrix} \begin{bmatrix} x(k) & x_{\mathrm{ref}}(k) & x_{\mathrm{se}}(k) \end{bmatrix} + u^{\mathrm{T}}(k) R\, u(k) \right] \tag{3.2.18}$$

kde Q_{se} je penalizace integrační akce.

Pokud se zvolí

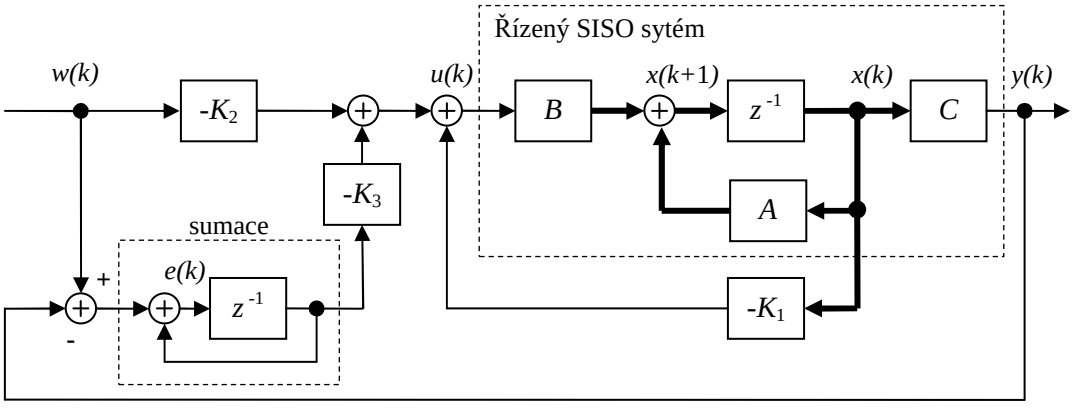
$$Q'' = \begin{bmatrix} C^{\mathrm{T}}QC & -C^{\mathrm{T}}Q & 0 \\ -QC & Q & 0 \\ 0 & 0 & Q_{\mathrm{se}} \end{bmatrix}, \tag{3.2.19}$$

lze řešit Riccatiho rovnici s použitím matic A'', B'' a Q'' místo matic A, B a Q. Výsledkem je matice zesílení K'' ve tvaru:

$$K'' = \begin{bmatrix} K_1 & K_2 & K_3 \end{bmatrix} \tag{3.2.20}$$

Optimální akční zásah se určí podle vztahu:

$$u(k) = -K_1 x(k) - K_2 x_{\mathrm{ref}}(k) - K_3 x_{\mathrm{se}}(k) = -K'' x''(k) \tag{3.2.21}$$



Obrázek 3.6: LQ regulátor s asymptotickým sledováním reference

Pro tento postup je nutné, aby byly všechny stavy měřitelné. Pokud existují neměřitelné stavy, musí se zavést rekonstruktor stavu.

3.3 PSEUDOSTAVOVÁ REPREZENTACE SYSTÉMU

Je daný diskrétní systém 3. řádu, který je popsaný následovně:

$$F(z)=\frac{b_1 z^{-1}+b_2 z^{-2}+b_3 z^{-3}}{1+a_1 z^{-1}+a_2 z^{-2}+a_3 z^{-3}} \tag{3.3.1}$$

$$y(k)=b_1 u(k-1)+b_2 u(k-2)+b_3 u(k-3)-a_1 y(k-1)-a_2 y(k-2)-a_3 y(k-3) \tag{3.3.2}$$

Stavová formulace nemusí být definovaná klasickým stylem – musí být pouze zachována matematická forma stavového modelu. Zavedeme proto nový stavový vektor $\bar{x}(k)$ složený ze zpožděných vstupů a výstupů soustavy:

$$\bar{x}(k)=\begin{bmatrix} u(k-1) \\ u(k-2) \\ u(k-3) \\ y(k-1) \\ y(k-2) \\ y(k-3) \end{bmatrix} \tag{3.3.3}$$

Nové matice $\overline{A}, \overline{B}, \overline{C}$ stavového systému jsou definovány:

$$\begin{aligned} \bar{x}(k+1) &= \overline{A}\bar{x}(k)+\overline{B}u(k) \\ y(k) &= \overline{C}u(k) \end{aligned} \tag{3.3.4}$$

Systém lze pomocí vztahu (3.3.2) přepsat následovně:

$$\begin{bmatrix} u(k) \\ u(k-1) \\ u(k-2) \\ y(k) \\ y(k-1) \\ y(k-2) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ b_1 & b_2 & b_3 & -a_1 & -a_2 & -a_3 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u(k-1) \\ u(k-2) \\ u(k-3) \\ y(k-1) \\ y(k-2) \\ y(k-3) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} u(k)$$

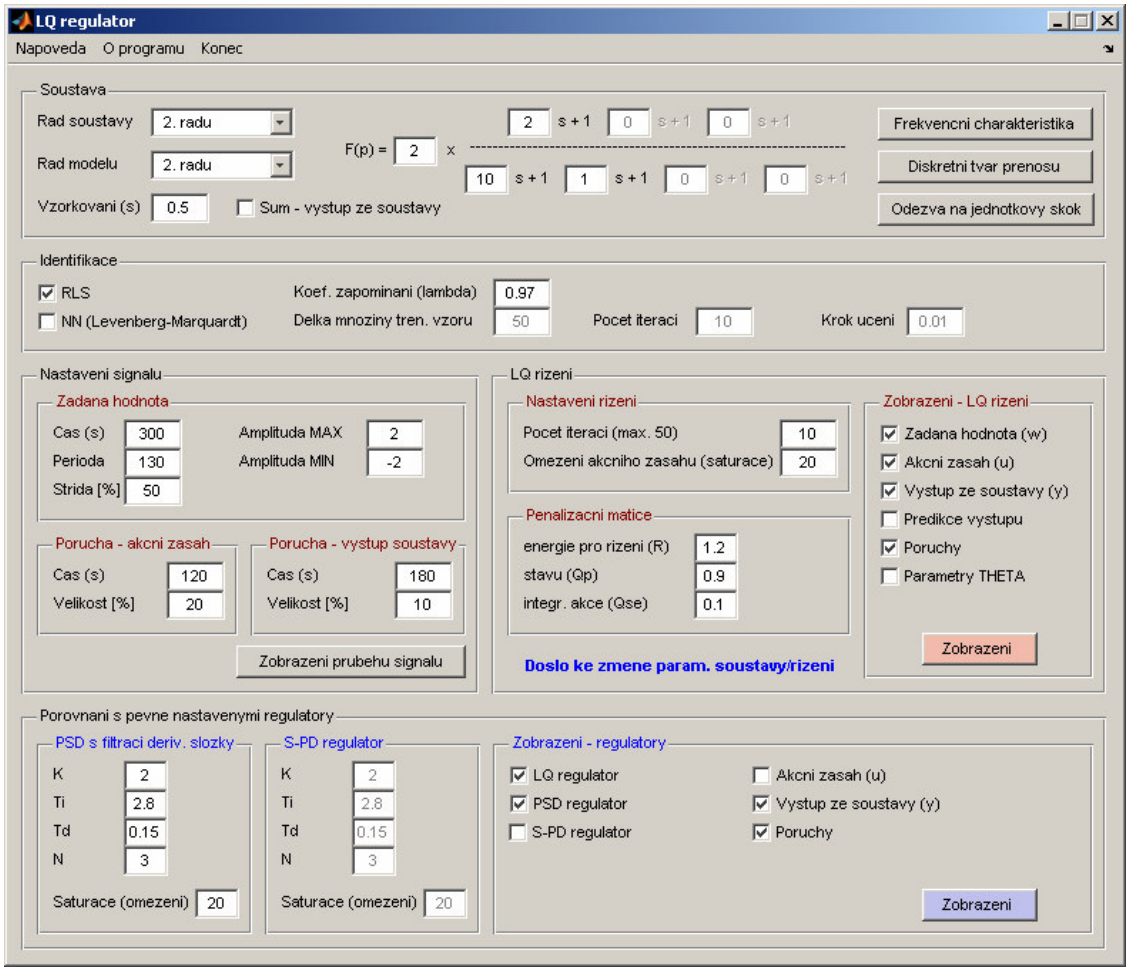
$$y(k) = \begin{bmatrix} b_1 & b_2 & b_3 & -a_1 & -a_2 & -a_3 \end{bmatrix} \begin{bmatrix} u(k-1) \\ u(k-2) \\ u(k-3) \\ y(k-1) \\ y(k-2) \\ y(k-3) \end{bmatrix} \qquad (3.3.5)$$

Výsledkem je tzv. pseudostavová reprezentace systému. Hodnoty modelu jsou ve skutečnosti z fyzikálního hlediska odlišné od systému, matematická reprezentace je však zachována – v algoritmu LQ řízení lze nahradit matice systému A, B, C novými maticemi $\overline{A}, \overline{B}, \overline{C}$.

4. PRAKTICKÉ OVĚŘENÍ VÝSLEDKŮ

4.1 PROGRAM „LQ REGULÁTOR“

Program „LQ regulátor“ slouží pro demonstraci adaptivního optimálního řízení. Výsledné průběhy lze porovnat s pevně nastaveným PSD a S-PD regulátorem.



Obrázek 4.1: Hlavní okno programu

Demonstrační program umožňuje zvolit:

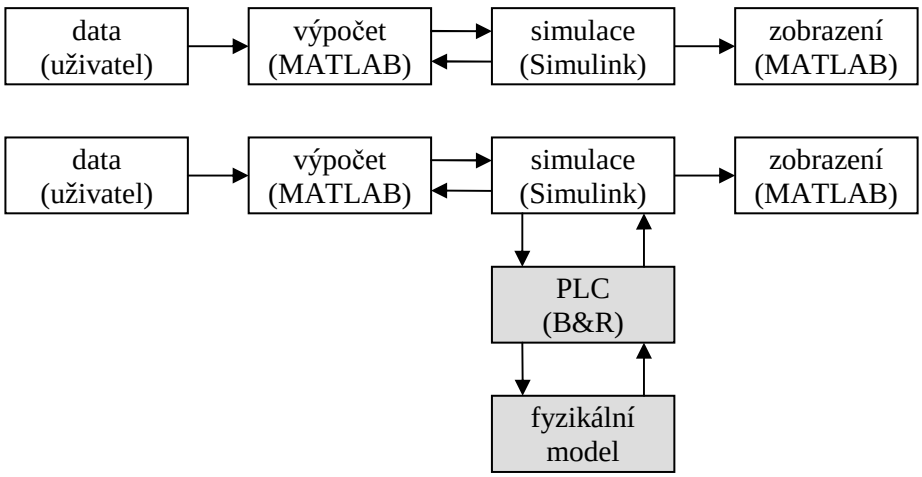
- libovolnou soustavu SISO 1. až 4. řádu
- řád modelu zadané soustavy pro identifikaci (1. až 4. řádu)
- metodu identifikace – metoda nejmenších čtverců / metoda Levenberg-Marquardt

- nastavení parametrů identifikace
- periodu vzorkování
- průběh žádané hodnoty (obdélníkový signál libovolné frekvence, amplitudy a střídy)
- velikost a čas působení poruchy žádané hodnoty a poruchy výstupu ze soustavy
- parametry optimálního kritéria
- koeficienty PSD a S-PD regulátorů (pro porovnání s LQ regulátorem)

Uživatel si dále může zvolit zobrazované průběhy. Přenos soustavy se zadává ve spojitém tvaru, parametry optimálního kritéria se zadávají jako určité poměry penalizační matice energie pro řízení (R), stavů (Q_p) a integrační akce (Q_{se}). Např. hodnoty $[0{,}5 \quad 0{,}3 \quad 0{,}7]$ jsou totožné s hodnotami $[5 \quad 3 \quad 7]$.

Program byl vytvořen ve 2 variantách:

- Soustava je simulována programem MATLAB/Simulink
- Soustava je realizována fyzikálním modelem, komunikace s PC probíhá pomocí PLC B&R



Obrázek 4.2: Varianty demonstračního programu LQ regulátor

V programu lze vybrat jednu z dvou metod identifikace – metodu nejmenších čtverců s exponenciálním zapomínáním (RLS) nebo metodu Levenberg-Marquardt (LM). Z metod identifikace pomocí neuronové sítě byla naprogramována metoda LM, protože rychleji konverguje k ustáleným hodnotám než metoda Back-propagation (BP) [16]. Krok učení LM metody je $\mu = 0{,}01$.

Identifikace probíhá v každém kroku. Na začátku řízení je po určitou dobu přiváděn na vstup soustavy konstantní signál (neprovádí se výpočet akčního zásahu). V tomto časovém intervalu se identifikuje soustava. Po určitém počtu kroků začne algoritmus počítat akční zásah podle zvoleného kritéria a získaných parametrů modelu (z identifikace). Během celé doby řízení je prováděna identifikace podle parametrů zadaných uživatelem.

Pozn.: Všechny skripty (pro program LQ regulátor a pro další simulace) byly psané pomocí skriptů M-file a S-function (resp. S-function s příkazy programu MATLAB).

4.1.1 Ukázka použití programu LQ regulátor

Zvolená soustava 2. řádu:

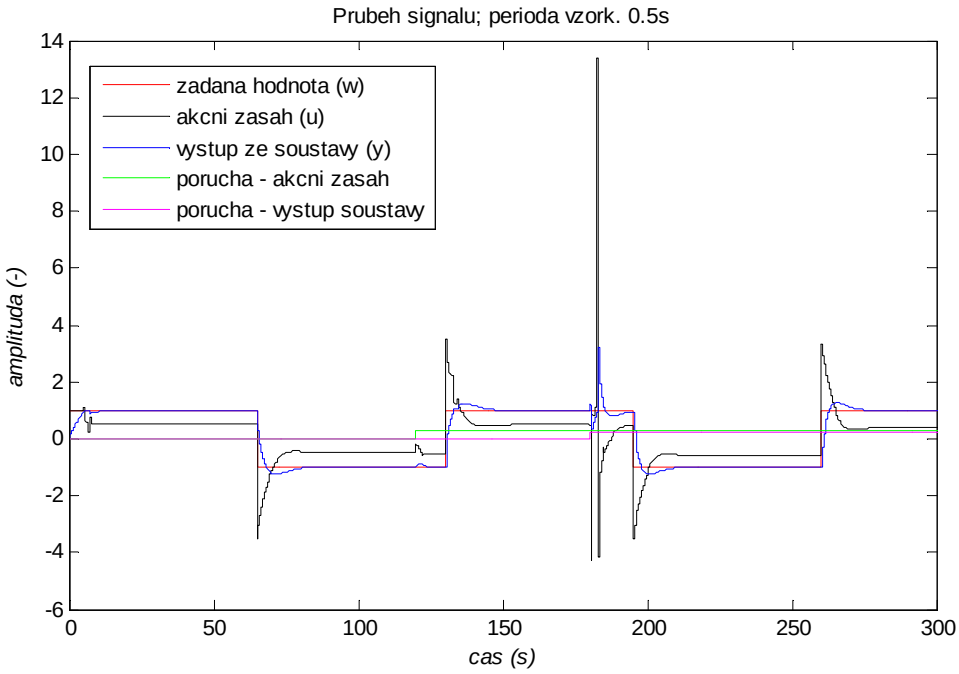
$$F(s) = \frac{2(2s+1)}{(10s+1)(s+1)} \qquad (4.1.1)$$

Identifikace: metoda nejmenších čtverců s koef. zapomínáním $\lambda = 0{,}97$, $T_S = 0{,}5\,\mathrm{s}$, model je 3. řádu. Penalizační matice byly zvoleny:

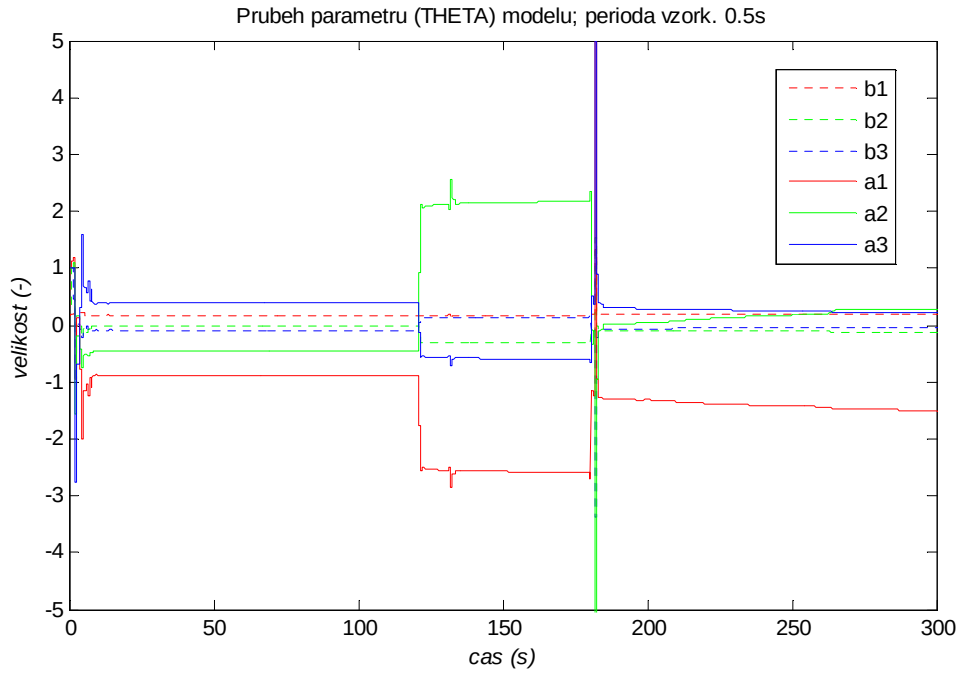
$R = 1{,}2$	energie pro řízení
$Q_p = 0{,}9$	stavů
$Q_{se} = 0{,}1$	integrační akce

Saturace akčního signálu byla nastavena na hodnotu +20/-20, amplituda žádané hodnoty byla rovna 1.

Dále byla přivedena porucha akčního zásahu o velikosti 30% amplitudy žádané hodnoty a porucha na výstup soustavy o velikosti 20%. Výsledné průběhy jsou na **Obrázku 4.3** a **Obrázku 4.4**.



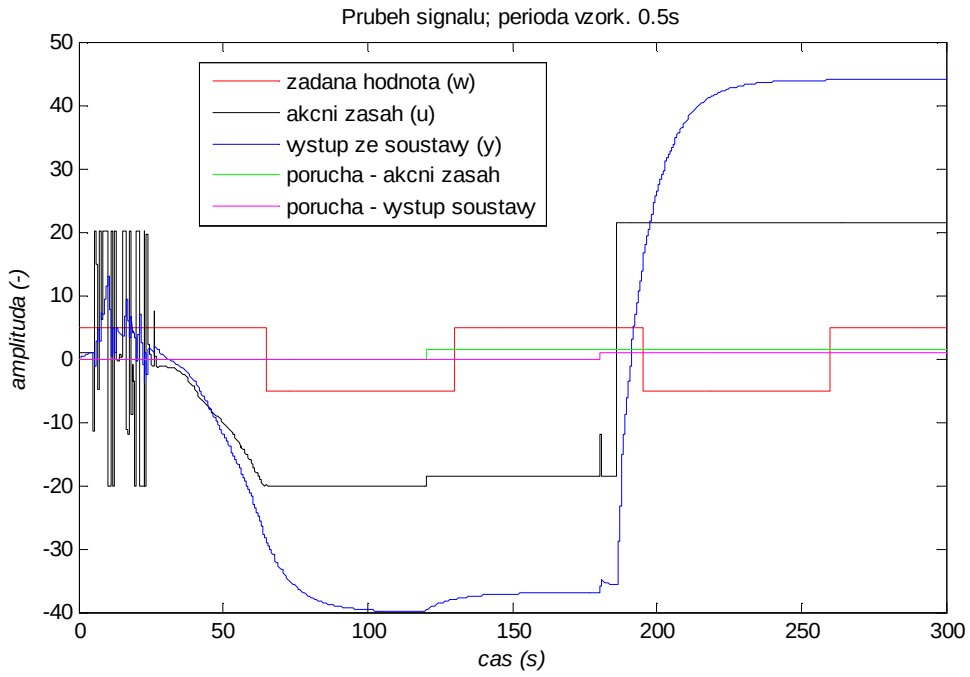
Obrázek 4.3: Ukázka průběhu signálů pro řízení zvolené soustavy



Obrázek 4.4: Ukázka průběhu parametrů modelu při řízení zvolené soustavy

Je důležité vhodně nastavit parametry LQ řízení. Na **Obrázku 4.5** jsou vidět průběhy signálů při řízení stejné soustavy (4.1.1). Byla použita stejná metoda identifikace i perioda vzorkování. Změna nastala pouze v penalizační matici energie pro řízení $R = 0{,}1$, amplituda žádané hodnoty na velikost 5.

Toto nastavení mělo za následek selhání LQ řízení způsobeného nedostatečnou velikostí akčního zásahu (omezený saturací), který byl penalizován menší vahou než v předchozím případě.

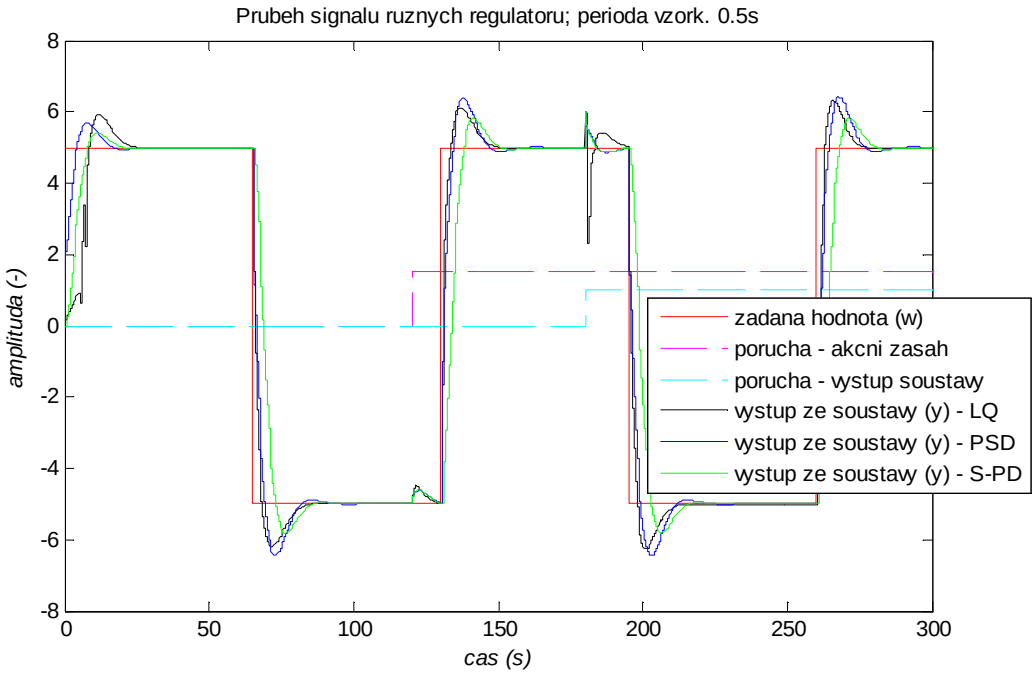


Obrázek 4.5: Případ selhání LQ řízení

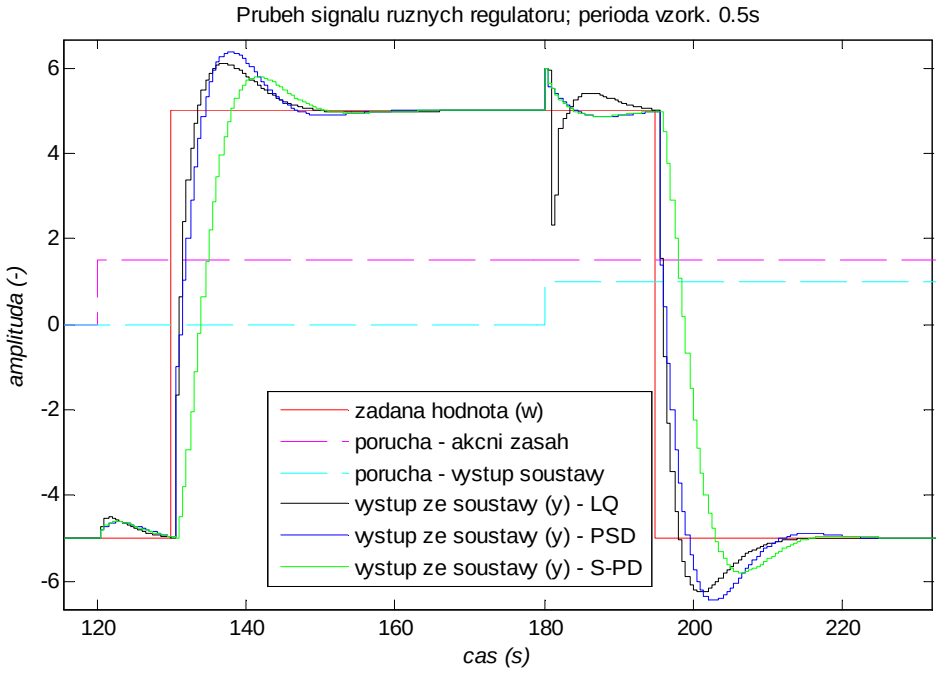
Další součástí demonstračního programu je porovnání LQ regulátoru s diskrétními ekvivalenty PID regulátorů. Program umožňuje zvolit parametry PSD a S-PD regulátoru a zobrazit průběhy signálů. Parametry je nutné volit na základě zkušeností (metoda pokus-omyl).

Zvolená soustava 2. řádu - stejná jako (4.1.1):

$$F(s) = \frac{2(2s+1)}{(10s+1)(s+1)} \qquad (4.1.2)$$



Obrázek 4.6: Porovnání LQ regulátoru s pevně nastav. diskr. ekvivalenty PID



Obrázek 4.7: Porovnání LQ reg. s pevně nastav. diskr. ekvivalenty PID (detail)

Identifikace: neuronová síť, metoda LM, $p = 50$, $\mu = 0{,}01$, $T_S = 0{,}5\,\mathrm{s}$, model je 3. řádu.

Penalizační matice byly zvoleny:

$R = 1{,}2$ energie pro řízení

$Q_p = 0{,}9$ stavů

$Q_{se} = 0{,}1$ integrační akce

Parametry PSD a S-PD regulátoru:

$K = 2$; $T_I = 2{,}8$; $T_D = 0{,}15$; $N = 2$; $saturace = 20$ (omezení shora i zdola)

Saturace akčního signálu byla nastavena na hodnotu +20/-20 (pro všechny regulátory), amplituda žádané hodnoty byla rovna 5.

Nastavení poruch:

porucha akčního zásahu - velikost 30% amplitudy žádané hodnoty

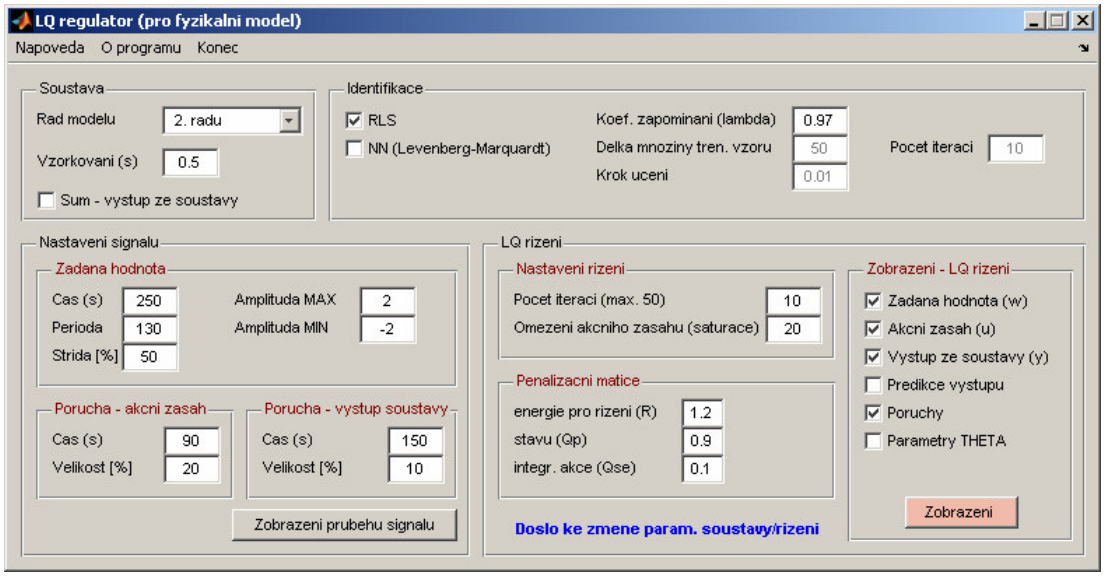
porucha na výstupu soustavy - velikost 20% amplitudy žádané hodnoty

Zobrazené průběhy jsou na **Obrázku 4.6** a **Obrázku 4.7** (detail). I po příchodu jednotlivých poruch vykazuje řízení soustavy s LQ regulátorem lepší průběh výstupní hodnoty (překmit a doba přechodu). Oproti PSD a S-PD regulátoru má LQ řízení horší odezvu na příchod poruchy na výstup soustavy – v identifikaci ze skokově změní parametry modelu, které krátkodobě způsobí špatný výpočet akčního zásahu LQ regulátoru. Díky zavedení integrační akce v algoritmu LQ řízení se porucha rychle vyreguluje.

4.1.2 Program LQ regulátor ve spojení s fyzikálním modelem

Další variantou demonstračního programu je nahrazení simulované soustavy skutečným fyzikálním modelem. V programu lze nastavit stejné parametry jako v předchozí variantě kromě parametrů PSD a S-PD regulátoru (kvůli délce simulace nelze provádět porovnávání řízení s PSD a S-PD regulátorem). Tento program není univerzální, protože komunikuje pouze s jediným automatem B&R. Pokud by byl

požadavek na komunikaci s jiným automatem, bylo by nutné změnit modelovací schémata, resp. změnit IP adresu zvoleného automatu.



Obrázek 4.8: Další varianta programu LQ regulátor (spojení s fyzik. modelem)

4.2 ŘÍZENÍ FYZIKÁLNÍHO MODELU

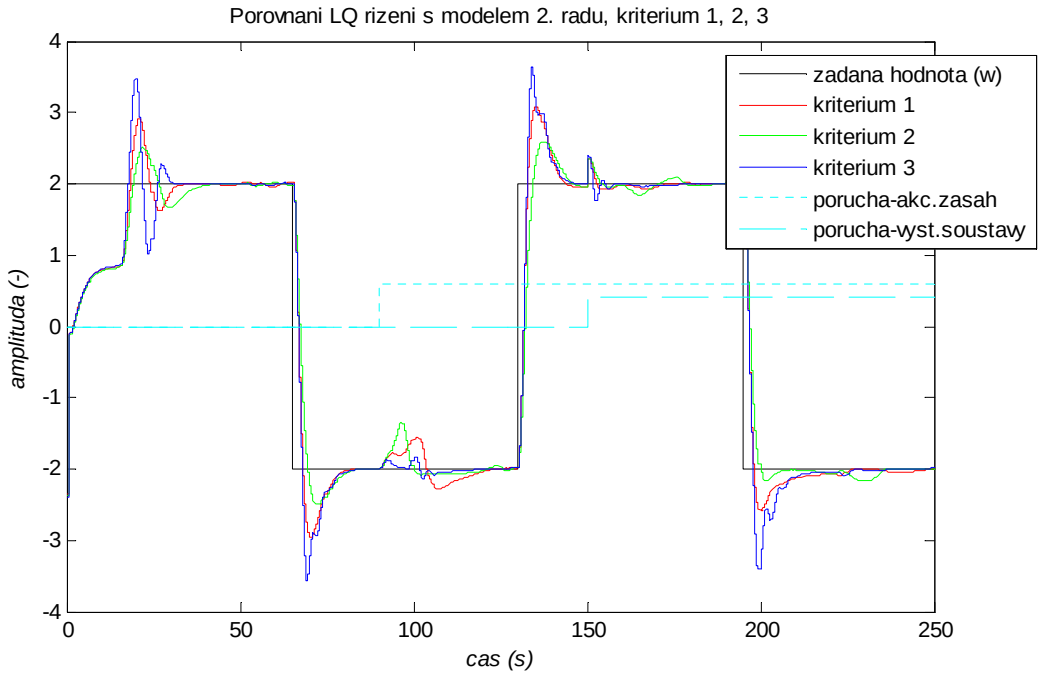
V této části byly ověřeny různé varianty řízení fyzikálního modelu – byl ověřován vliv volby řádu soustavy, parametrů optimálního kritéria, šum na výstupu soustavy atd.

Penalizační matice	kritérium 1	kritérium 2	kritérium 3
R (energie pro řízení)	0,5	1,2	0,5
Q_p (stavů)	0,9	0,9	0,9
Q_{se} (integrační akce)	0,1	0,5	0,5

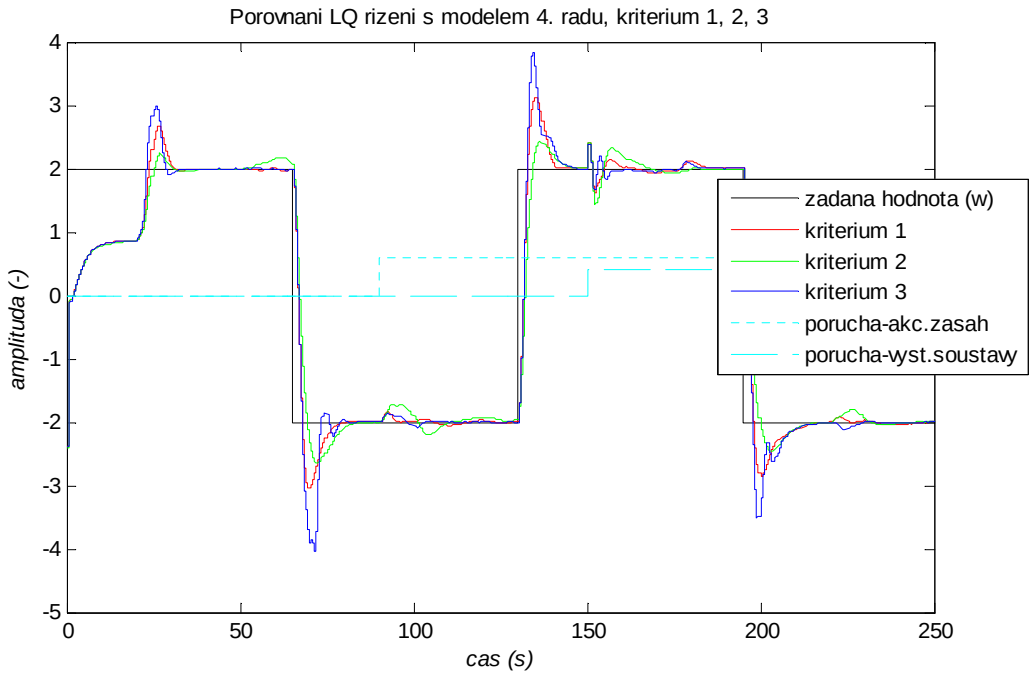
Tabulka 4.1: Nastavené penalizační matice pro různá kritéria

Identifikace byla prováděna vždy metodou LM s nastavením $p = 50$, $\mu = 0{,}01$ při saturaci akčního signálu nastaveného na hodnotu +20/-20. Amplituda žádané hodnoty byla rovna 2. Poruchy byly nastaveny opět stejně: porucha akčního

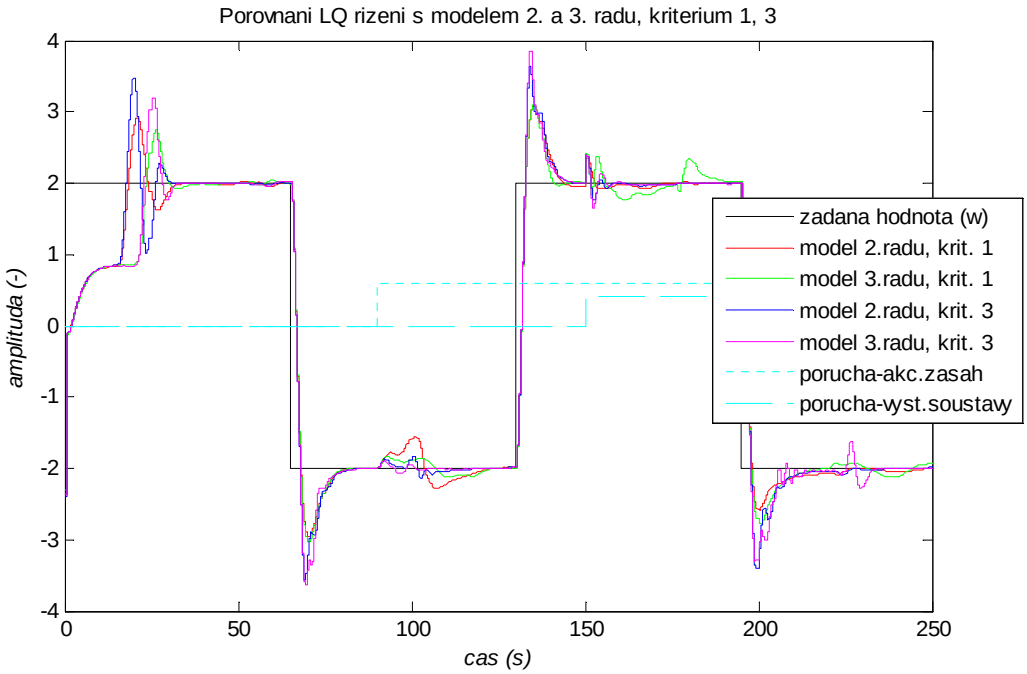
zásahu 30% amplitudy žádané hodnoty a porucha na výstupu soustavy 20% amplitudy žádané hodnoty. Perioda vzorkování byla $T_S = 0{,}5\text{s}$.



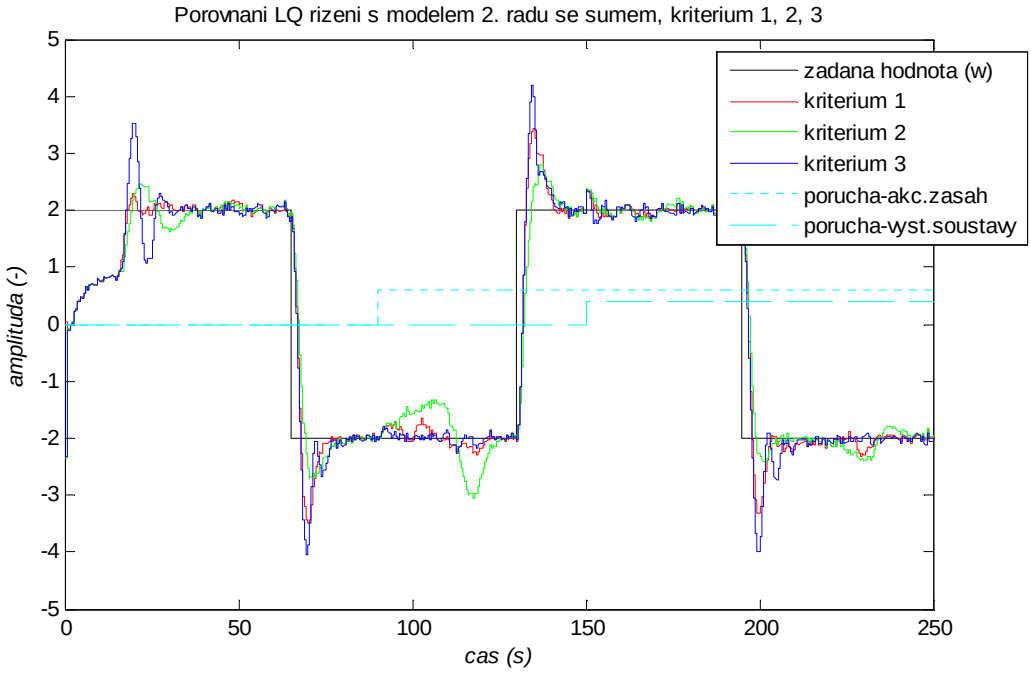
Obrázek 4.9: Porovnání LQ řízení s modelem 2. řádu, kritérium 1, 2, 3



Obrázek 4.10: Porovnání LQ řízení s modelem 4. řádu, kritérium 1, 2, 3



Obrázek 4.11: Porovnání LQ řízení s modelem 2. a 3. řádu, kritérium 1, 3



Obrázek 4.12: Porovnání LQ řízení s modelem 2. a 3. řádu se šumem, krit. 1, 2, 3

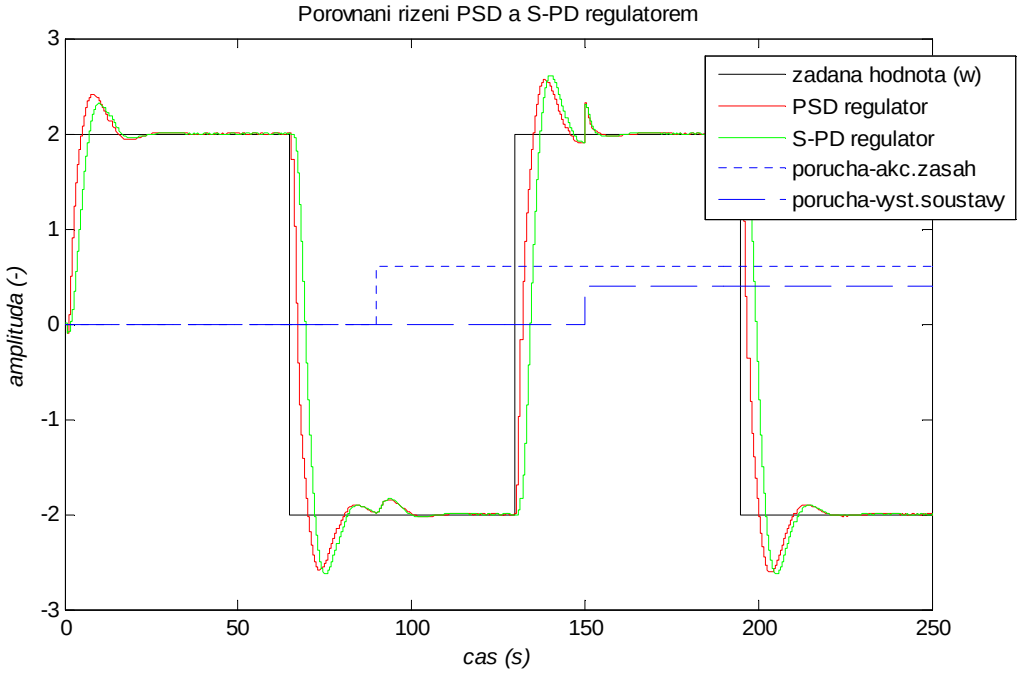
Z uvedených grafů vyplývá, že průběh výstupní hodnoty závisí na řádu modelu a na parametrech optimálního kritéria. Nelze tedy říci, které nastavení by bylo optimální pro každou soustavu. Dalším důležitým parametrem je nastavení identifikace. Pokud identifikační algoritmus (resp. nastavení) pomalu reaguje na působící poruchy, může to vést k selhání řízení.

Další část ověřování je zaměřena na porovnání řízení fyzikálního modelu s pevně nastavenými diskrétními ekvivalenty PID.

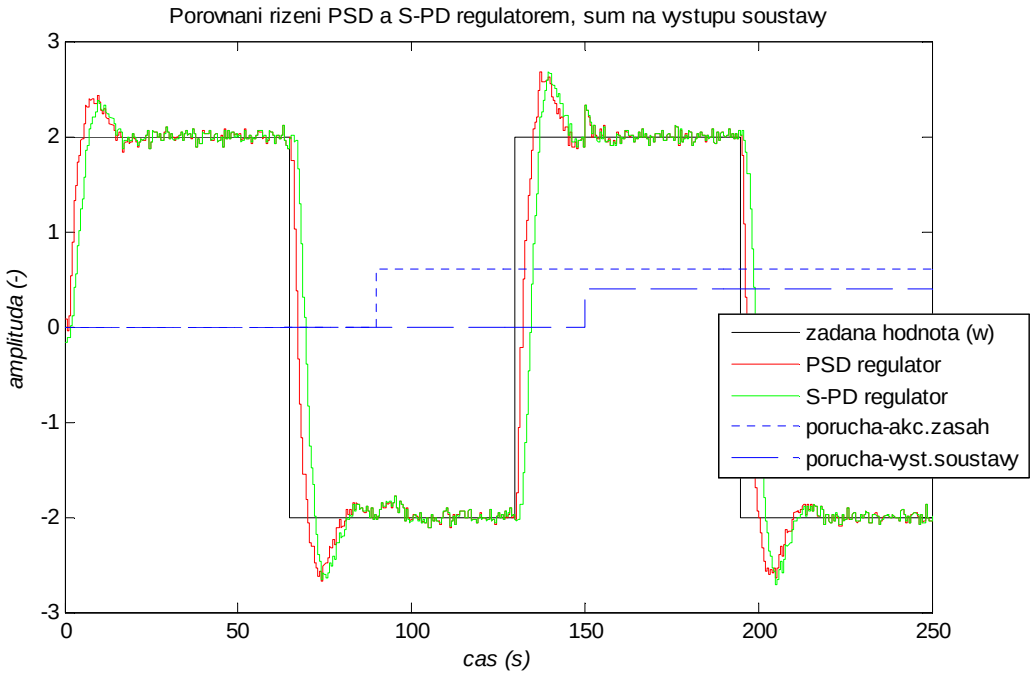
Nastavení PSD a S-PD bylo následující (pro oba regulátory stejné):

$K = 1{,}5\ ;\ T_{\mathrm{I}} = 2{,}1\ ;\ T_{\mathrm{D}} = 1{,}1\ ;\ N = 2\ ;\ saturace = 20$ (omezení shora i zdola)

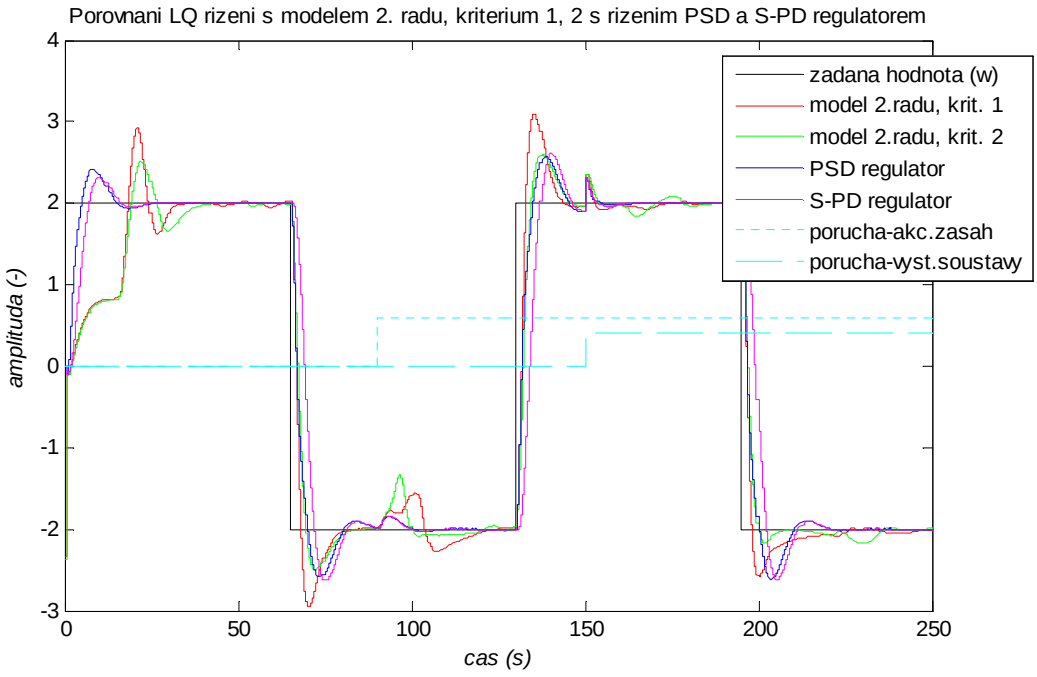
Z výsledných grafů plyne, že při nevhodné volbě parametrů optimálního kritéria nebo řádu modelu vychází horší průběhy než při řízení pomocí klasického PSD regulátoru.



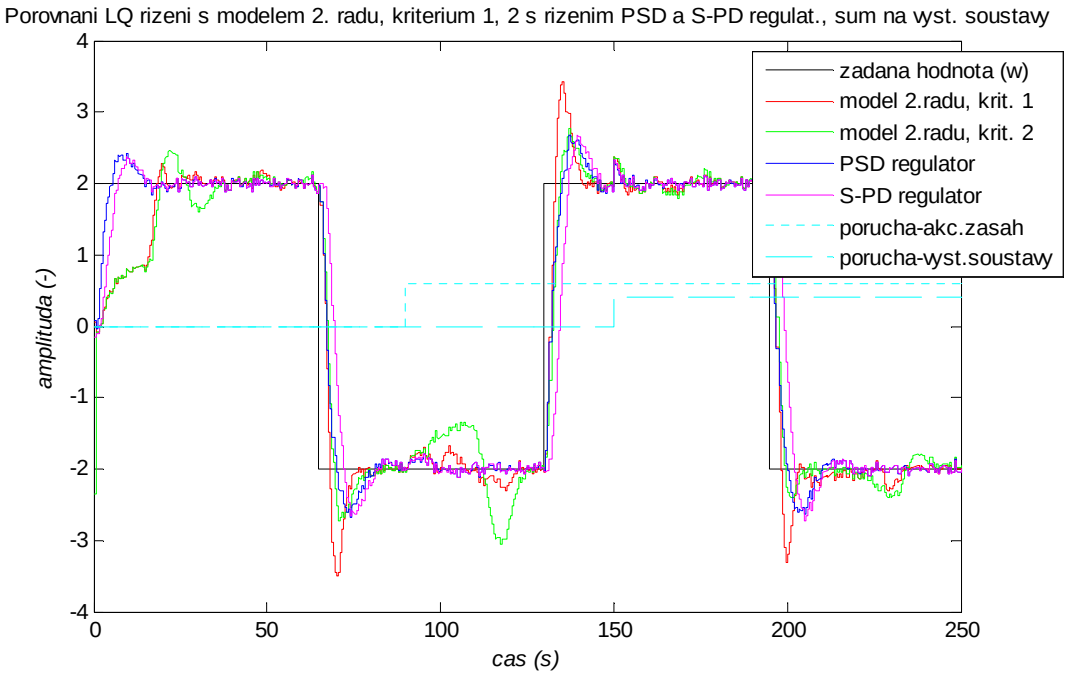
Obrázek 4.13: Porovnání řízení PSD a S-PD regulátorem



Obrázek 4.14: Porovnání řízení PSD a S-PD regulátorem, šum na výst. soustavy



Obrázek 4.15: Porovnání LQ řízení s mod. 2. řádu, krit. 1, 2 s PSD a S-PD



Obrázek 4.16: Porovnání LQ řízení s mod. 2. řádu, krit. 1, 2 s PSD a S-PD + šum

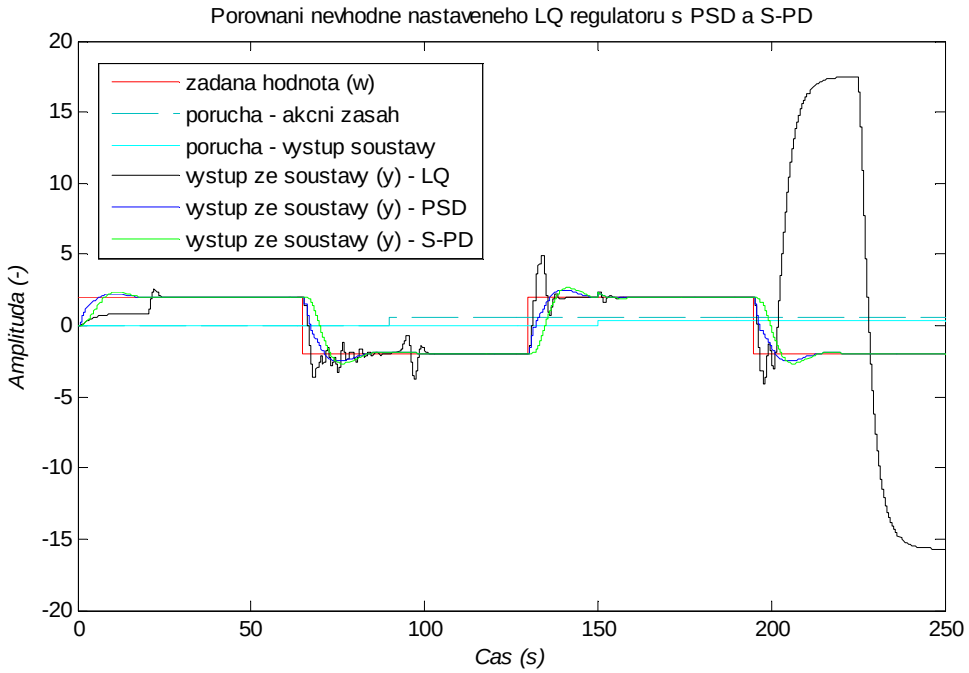
Na **Obrázku 4.17** je vidět situace, kdy LQ řízení selhává oproti řízení pomocí PSD regulátoru. Selhání řízení bylo způsobeno nevhodným nastavením parametrů kritéria:

$R = 0{,}1$	penalizační matice energie pro řízení
$Q_{\mathrm{p}} = 0{,}9$	penalizační matice stavů
$Q_{\mathrm{se}} = 0{,}9$	penalizační matice integrační akce

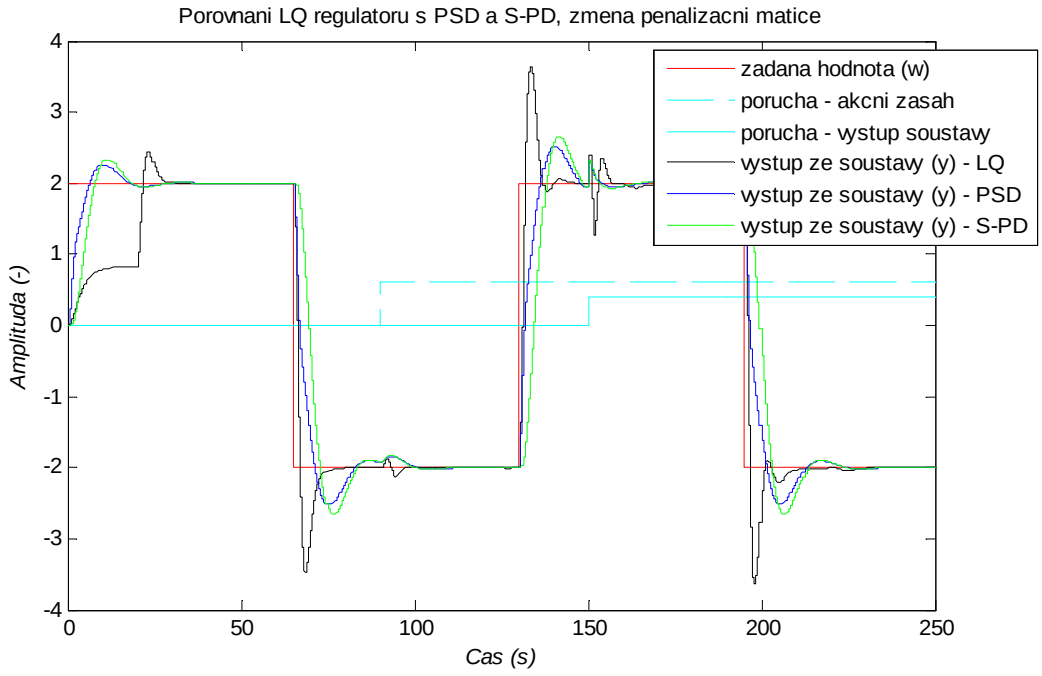
Identifikace byla prováděna metodou LM s nastavením $p = 50$, $\mu = 0{,}01$. Saturace akčního signálu byla nastavena na hodnotu +20/-20 (pro všechny regulátory), amplituda žádané hodnoty byla rovna 2. Poruchy byly nastaveny opět stejně: porucha akčního zásahu 30% amplitudy žádané hodnoty a porucha na výstupu soustavy 20% amplitudy žádané hodnoty. Model soustavy byl 3. řádu a perioda vzorkování $T_{\mathrm{S}} = 0{,}5\mathrm{s}$.

Nastavení PSD a S-PD bylo následující (pro oba regulátory stejné):

$K = 1{,}5$; $T_{\mathrm{I}} = 2{,}1$; $T_{\mathrm{D}} = 1{,}1$; $N = 2$



Obrázek 4.17: Porovnání nevhodně nastaveného LQ regulátoru s PSD a S-PD



Obrázek 4.18: Porovnání LQ regulátoru (změna penalizace energie) s PSD a S-PD

Je vidět, že energie pro řízení není vůbec penalizována. Proto se na vstup soustavy přivádí velké změny akčního zásahu, které způsobí, že dojde k rozkmitání (v **Obrázku 4.17** není kvůli přehlednosti zobrazen akční zásah). Pokud se zvolí penalizační matice energie $R = 0{,}9$, akční zásahy nejsou již tak velké a řízení se stane stabilní (**Obrázek 4.18**).

4.3 VOLBA IDENTIFIKACE V ZÁVISLOSTI NA PRŮBĚHU ŘÍZENÍ

Identifikace má velký vliv na průběh řízení. Ověřování probíhalo na soustavě s přenosem

$$F(s) = \frac{1}{(4s+1)(s+1)} \qquad (4.3.1)$$

Identifikace probíhaly se vzorkováním $T_S = 0{,}5\,\mathrm{s}$ byly nastaveny:

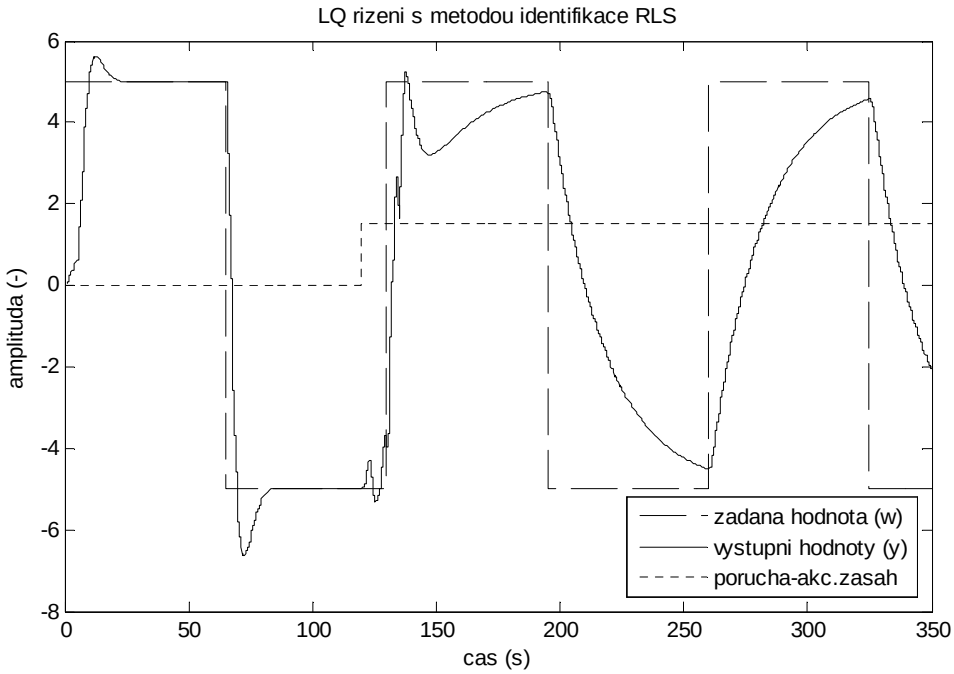
RLS: $\lambda = 0{,}97$

LM: $p = 50$; $\mu = 0{,}01$

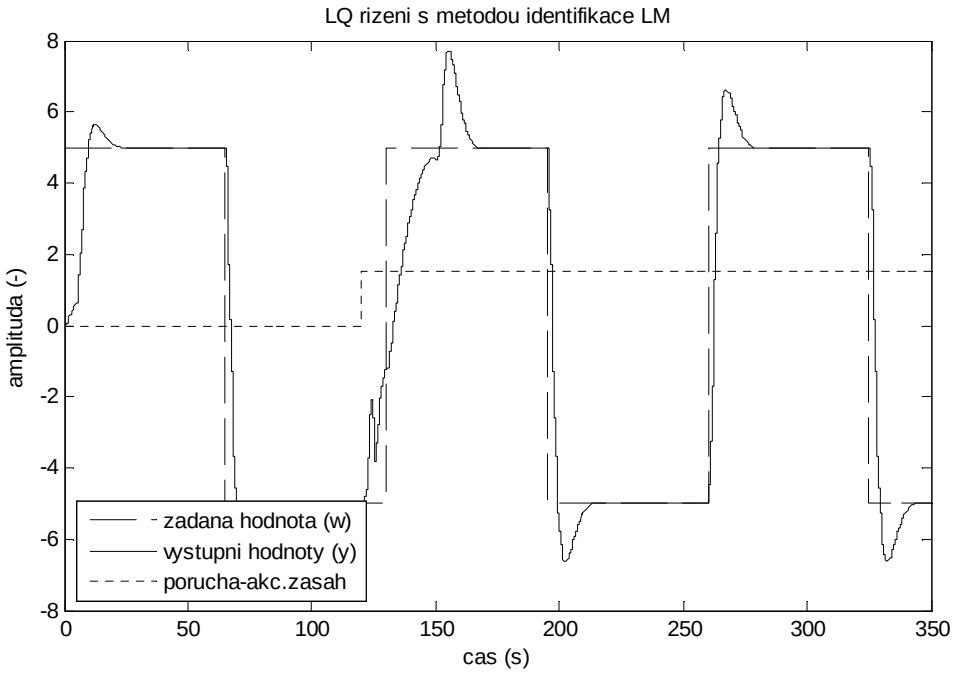
Penalizační matice byly zvoleny: $R = 0{,}9$; $Q_p = 0{,}5$; $Q_{se} = 0{,}1$

Amplituda žádané hodnoty měla velikost 5 a porucha akčního zásahu měla hodnotu 30% amplitudy žádané hodnoty. V obou případech byl zvolený model 3. řádu.

Z průběhů na **Obrázku 4.19** a **Obrázku 4.20** je vidět, že porucha akčního zásahu přišla v nepříznivý okamžik – došlo ke skokové změně žádané hodnoty v okamžiku, kdy ještě nebyla vyregulována porucha akčního zásahu. Identifikace pomocí metody RLS selhala – LQ regulátor dostal zavádějící informace o řízené soustavě. Naopak LQ řízení s identifikací metodou LM po určité době poruchu vyregulovalo a výstupní signál měl stejný průběh jako před příchodem poruchy.



Obrázek 4.19: LQ řízení s metodou identifikace RLS



Obrázek 4.20: LQ řízení s metodou identifikace LM

4.4 ZMĚNA DYNAMIKY SOUSTAVY

Pevně nastavený PSD regulátor není schopen sám se přizpůsobit dané soustavě. Pokud je změna dynamiky soustavy malá, projeví se to méně příznivějším průběhem výstupní hodnoty. Pokud bude ale tato změna velká, může dojít k selhání řízení.

V této části byl ukázán průběh signálů optimální řízení při změně dynamiky soustavy. Průběhy byly porovnány s průběhy řízení pomocí PSD a S-PD regulátorů.
Zvolené nastavení:

soustava 1 s přenosem $F(s)=\frac{2s+1}{10s^2+4s+11}$ (4.4.1)

soustava 2 s přenosem $F(s)=\frac{2s+1}{10s^2+11s+11}$ (4.4.2)

čas změny $t=150$ s , perioda vzorkování $T_S=0{,}5$ s

identifikace: metoda LM, $p=50$, $\mu=0{,}01$, model 2. řádu

$R=1{,}2$ penalizační matice energie pro řízení

$Q_p=0{,}9$ penalizační matice stavů

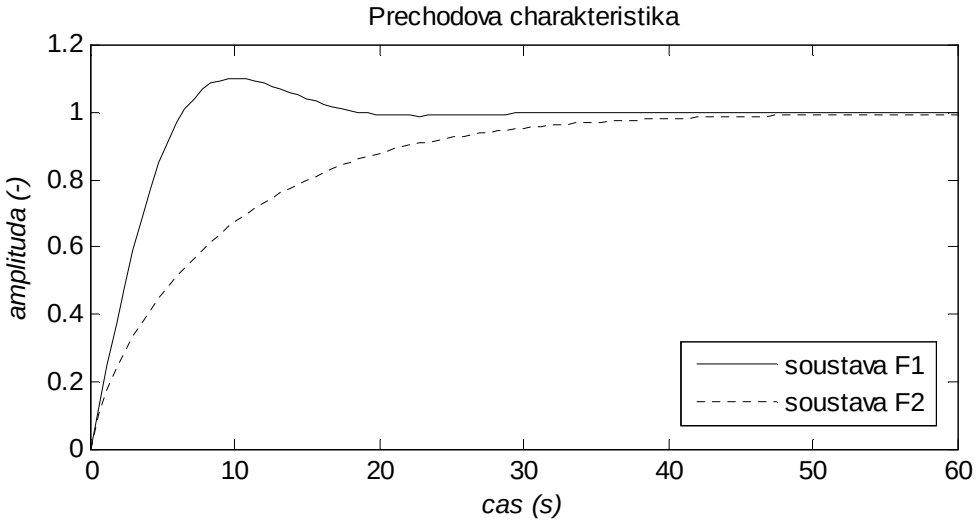
$Q_{se}=1$ penalizační matice integrační akce

Nastavení PSD: $K=1{,}5$; $T_I=2{,}8$; $T_D=1{,}1$; $N=2$

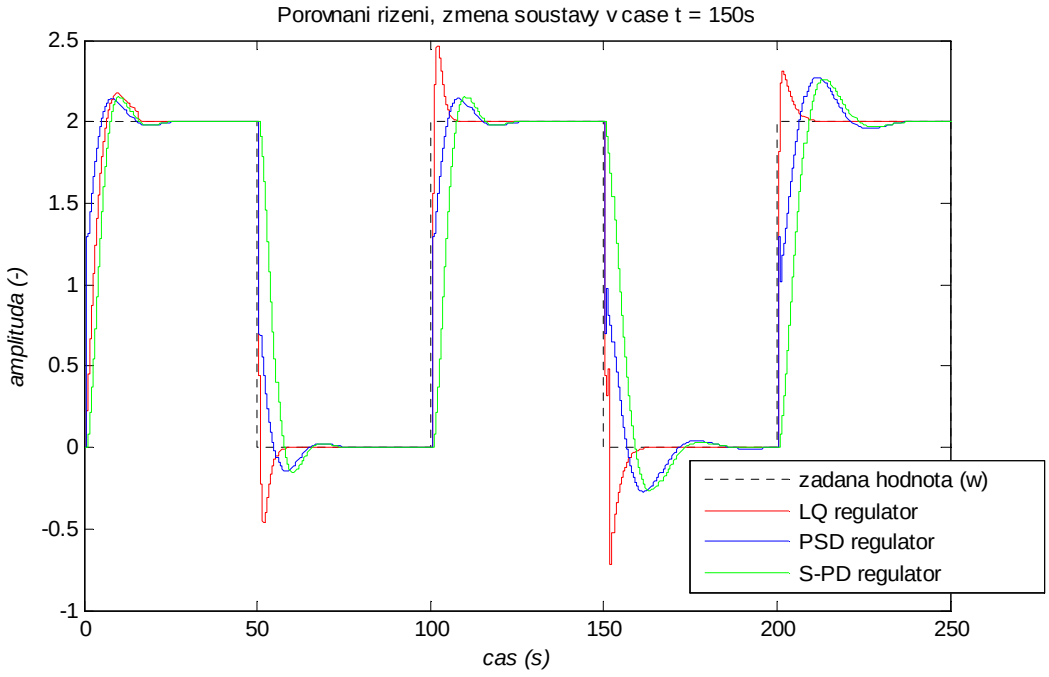
Nastavení S-PD: $K=1{,}5$; $T_I=2{,}8$; $T_D=0{,}9$; $N=2$

Saturace byla pro všechny regulátory nastavena na hodnotu +20/-20.

V tomto případě musí být splněn jeden základní předpoklad: adaptivní řízení bude probíhat správně, pokud se správně identifikuje soustava. To nastane, když je na vstup přiváděn dostatečně „bohatý“ budicí signál (např. jednotkový skok nebo bílý šum). Výše uvedená podmínka byla splněna – v čase $t=150$ s došlo ke změně soustavy a zároveň došlo i ke změně žádané hodnoty.



Obrázek 4.21: Odezva na jednotkový skok soustav F_1 a F_2



Obrázek 4.22: Porovnání řízení při změně dynamiky soustav v čase t = 150 s

Nejméně příznivé průběhy byly dosaženy při řízení S-PD regulátorem. To je způsobené jinou strukturou vnitřního zapojení oproti PSD regulátoru - do proporcionální a diferenční složky není přiváděna odchylka $e(k)$, ale záporná hodnota výstupu ze soustavy $-y(k)$.

4.5 LINEÁRNĚ NARŮSTAJÍCÍ PRŮBĚH ŽÁDANÉ HODNOTY

Bylo ověřeno, jak se chová LQ regulátor, pokud je žádanou hodnotou lineárně rostoucí funkce s omezením (tvar rampy). Z výsledných hodnot je vidět, že výstup soustavy opisuje žádanou hodnotu lépe při použití optimálního řízení. Na druhou stranu při použití LQ řízení dochází k většímu překmitu. Na **Obrázku 4.24** je ukázán průběh signálů, pokud na soustavu působí poruchy ve tvaru rampy. Nejméně příznivý průběh výstupní hodnoty nastal při použití S-PD regulátoru.

Nastavení bylo následující:

Soustava: $$F(s) = \frac{0{,}8}{(3s+1)(s+1)} \qquad (4.5.1)$$

Perioda vzorkování: $T_S = 0{,}5$ s

LQ regulátor:

identifikace: metoda LM, $p = 50$, $\mu = 0{,}01$, model 2. řádu

$R = 1{,}2$ penalizační matice energie pro řízení

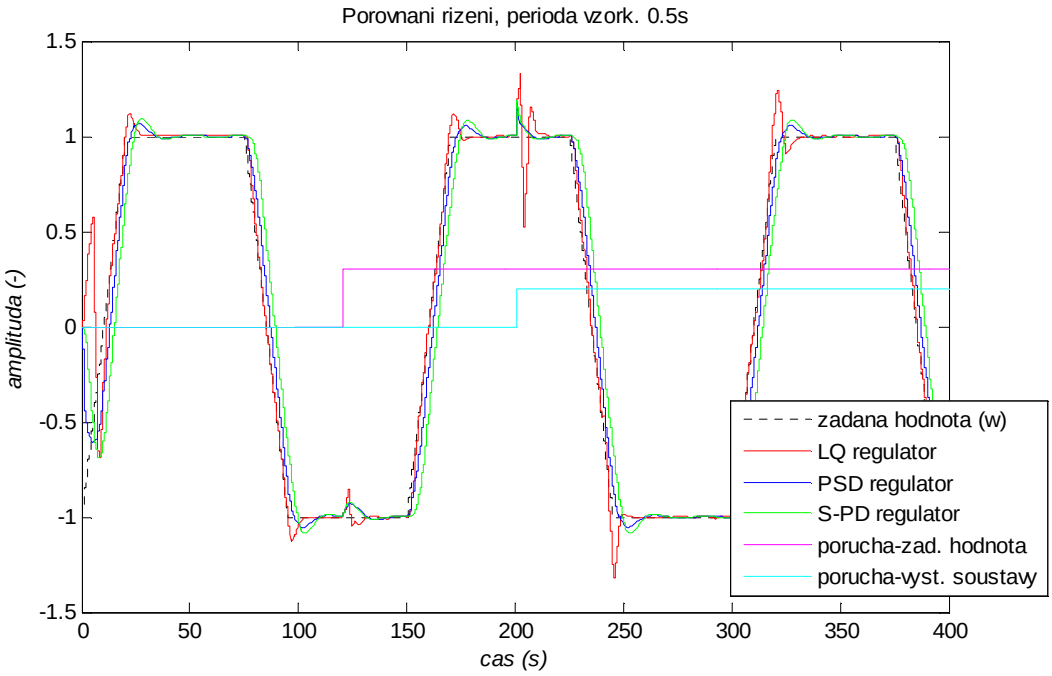
$Q_p = 0{,}9$ penalizační matice stavů

$Q_{se} = 1$ penalizační matice integrační akce

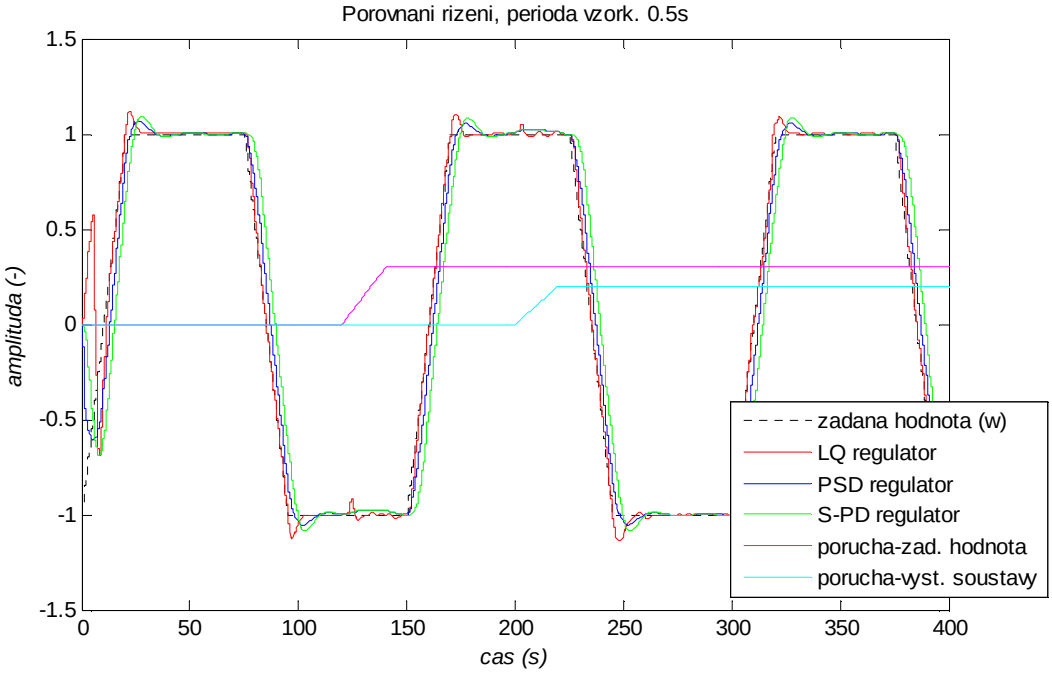
Amplituda žádané hodnoty měla velikost 1. Porucha akčního zásahu měla hodnotu 30% amplitudy žádané hodnoty a porucha působící na výstupu soustavy měla hodnotu 20% amplitudy žádané hodnoty. Saturace byla nastavena pro všechny regulátory na hodnotu +20/-20.

Nastavení PSD a S-PD bylo následující (pro oba regulátory stejné):

$K = 1{,}5$; $T_I = 2{,}1$; $T_D = 1{,}1$; $N = 2$



Obrázek 4.23: Porovnání řízení, žádaná hodnota ve tvaru rampy, skoková porucha



Obrázek 4.24: Porovnání řízení, žádaná hodnota a porucha ve tvaru rampy

5. PROGRAMOVATELNÉ AUTOMATY (PLC)

Programovatelné automaty (PLC – Programmable Logic Controller) slouží pro řízení technologických procesů v automatizaci. Výhodnou PLC oproti řídicím počítačům je [17]:

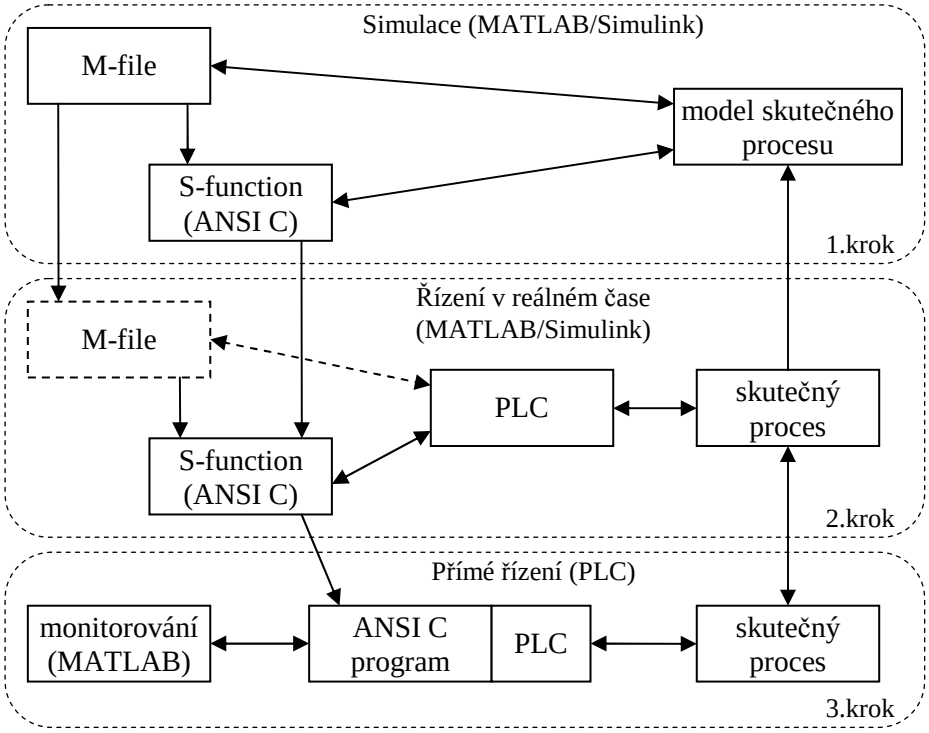
- spolehlivost
- snazší rozdělení řídicí struktury na samostatné celky s jasně definovatelnými rozhraními
- nižší náklady na kabeláž
- modularita
- vestavěná diagnostika vlastního PLC
- rychlejší uvedení do chodu
- jednodušší údržba, ladění programů apod.

5.1 IMPLEMENTACE ALGORITMU DO PLC

Postup implementace algoritmu do PLC probíhá ve třech krocích [8]:

- krok 1: vytvoření algoritmu, upravování a testování pomocí simulačních programů (MATLAB/ Simulink)
- krok 2: algoritmus je testován na skutečném procesu; MATLAB/Simulink zajišťuje vykonávání programu, komunikaci mezi PC a skutečným procesem zajišťuje programovatelný automat (PLC)
- krok 3: vytvořený a odladěný algoritmus je implementovaný přímo do paměti programovatelného automatu; PLC sám autonomně řídí skutečný proces, MATLAB/Simulink (nebo jiný vhodný program) zajišťuje pouze vizualizaci průběhu řízení

Tento postup (ve správném pořadí) by měl zajistit, aby implementace byla co nejméně pracná a aby se největší množství chyb odladilo již při simulaci.



Obrázek 5.1: Blokové schéma implementace algoritmu do PLC [8]

5.2 PROGRAMOVACÍ JAZYK

Existují různé typy programovacích jazyků PLC definovaných normou IEC1131-3. Jsou to Ladder Diagram (LD), Instruction List (IL), Structured Text (ST), Sequential Function Chart (SCF) a jazyk C, resp. jeho standardizovaná verze ANSI C. V případě implementace složitějších algoritmů je výhodné použít právě jazyk ANSI C. Výhodou je přenositelnost mezi automaty různých výrobců. V programu MATLAB lze vytvořit algoritmus v jazyce ANSI C ve tvaru tzv. S-function, kterou lze přeložit do dynamicky linkované knihovny (DLL). S touto knihovnou pracuje Simulink rychleji než s klasickou funkcí tvořenou příkazy programu MATLAB (tzv. M-file S-function).

Algoritmus pro řízení procesu je nejdříve vyvíjen v simulačním prostředí. Po odladění zásadních chyb lze algoritmus přímo implementovat do PLC (krok 3). Algoritmus je potom přímo uložen v PLC a MATLAB/Simulink zde může (ale i nemusí) sloužit jako pozorovatel procesu. Dále může vykreslovat průběhy signálů nebo měnit žádanou hodnotu.

5.3 SPOJENÍ MATLAB/SIMULINK – PLC

Pro komunikaci mezi počítačem a PLC firmy B&R se využívá softwarové rozhraní PVI (Process Visualization Interface). Při této komunikaci se používá sériová linka RS232, rychlý Ethernet, Profibus, CAN nebo i Modem.

Komunikační protokol Ethernet TCP/IP je nedeterministický. Proto byla vytvořena nová komunikační knihovna (Real-Time Tolbox) v prostředí MATLAB/Simulink místo stávajícího standardního řešení pomocí knihoven Real-Time Workshop a Real-Time Windows Target. Důvody pro vytvoření nové komunikační knihovny:

- neexistující komunikace pracující v reálném čase mezi prostředím MATLAB/Simulink a průmyslovým regulátorem PLC, která je stabilní, robustní a její použití je pro uživatele jednoduché
- navrhované řešení můžeme považovat za komplexní pro přímou implementaci řídicích algoritmů, byla tak vytvořena klíčová druhá část pro přímou implementaci řídicích algoritmů

Operační systémy Windows NT, 2000 nebo XP, na kterých se MATLAB/Simulink používá, patří do skupiny Soft Real-Time. I když mají tyto operační systémy implementované různé doplňky (knihovny), nepatří k operačním systémům pevného reálného času (Hard Real-Time). Do této kategorie patří právě programovatelné průmyslové automaty PLC [1].

5.3.1 Reakční doba komunikace s PLC

Reakční doba komunikace s PLC se dělí na následující složky [8]:

1) čtení výstupní hodnoty procesu z PLC (T_1)
2) přenos výstupní hodnoty procesu do MATLAB/Simulink (T_2)
3) výpočet akčního zásahu (vstup do procesu) v MATLAB/Simulink ze známé žádané hodnoty a výstupní hodnoty procesu(T_3)
4) přenos vstupní hodnoty procesu (akční zásah) z MATLAB/Simulink (T_4)
5) zápis vstupní hodnoty procesu do PLC (T_5)

Celková reakční doba je dána: $T = T_1 + T_2 + T_3 + T_4 + T_5$

6. ZÁVĚR

V práci byly popsány různé metody identifikace. Při ověřování těchto metod se nezjistily podstatné rozdíly v kvalitě identifikace. U každé metody záleželo především na nastavených parametrech. Dalším důležitým hlediskem byl zvolený řád modelu. Z pohledu náročnosti výpočtu se ukazuje jako výhodnější metoda nejmenších čtverců – jedná se o iterační metodu (malá náročnost na paměť) a není nutné určovat inverzi matice. Naopak výpočetně náročnější metoda Levenberg-Marquardt nejrychleji konverguje k ustáleným hodnotám. Proto se tato metoda ukazuje výhodnější při ověřování na fyzikální modelu.

LQ řízení je typ takového řízení, které minimalizuje kvadratické optimální kritérium. Akční zásah je roven skalárnímu součinu koeficientů (proporcionálních složek) zpětných vazeb a jednotlivých stavů systému. Pokud je k dispozici správný model soustavy (získaný identifikací), lze LQ regulátor řešit jako adaptivní LQ regulátor.

Obecně popsaný optimální regulátor řeší úlohu pro minimální vektor stavu, tj. snaží se dostat všechny stavy do nuly. Problém žádané hodnoty se řeší úpravou struktury optimalizačního algoritmu. Optimální regulátor neřeší problém trvalé ustálené odchylky. Je proto nutné do algoritmu zahrnout i integrační složku.

Dále je důležité si uvědomit, že pomocí LQ řízení získáme nejlepší řešení podle daného kritéria. Toto řešení ale nemusí být výhodné z pohledu uživatele. Např. běžný PSD regulátor, který bude mít horší vlastnosti vzhledem k danému kritériu, může být z pohledu uživatele vhodnější pro použití k řízení soustavy.

Součástí diplomové práce bylo vytvoření programu, pomocí kterého lze názorně demonstrovat vlastnosti LQ řízení. Algoritmus může ovšem v praxi selhávat v případě špatné identifikace soustavy. Tento případ nastane při nevhodném průběhu budicího signálu (signál není dostatečně „bohatý“).

7. SEZNAM LITERATURY

[1] PIVOŇKA, P.: *Číslicová řídicí technika*. VUT Brno, skriptum, 2003.

[2] BOBÁL, V. - BÖHM, J. - PROKOP, R. - FESSL, J.: *Praktické aspekty samočinně se nastavujících regulátorů: algoritmy a implementace.* VUTIUM, 1999. ISBN 80-214-1299-2.

[3] ZELINKA, I.: *Umělá inteligence 1 – Neuronové sítě a genetické algoritmy.* VUTIUM, 1998. ISBN 80-214-1163-5.

[4] ONDRÁČEK, T.: *Adaptivní vícevrstvé neuronové sítě.* Edice PhD Thesis, sv. 361, VUT Brno, 2006. ISBN 80-214-3126-1. ISSN 1213-4198.

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

[6] DRÁBEK, O. - TAUFER, I. - SEIDL, P.: Umělé neuronové sítě – základy teorie a aplikace (3). *CHEMagazín*. 2006, roč. XVI, č. 1, str. 12-14. ISSN 1210-7409.

[7] DRÁBEK, O. - TAUFER, I. SEIDL, P.: Umělé neuronové sítě – základy teorie a aplikace (5). *CHEMagazín*. 2006, roč. XVI, č. 5, str. 29-31. ISSN 1210-7409.

[8] ŠVANCARA, K.: *Adaptive optimal controller with identification based on neural networks.* Brno, 2004. Disertační práce na Fakultě elektrotechniky a komunikačních technologií Vysokého učení technického v Brně na Ústavu automatizace. Vedoucí disertační práce Prof. Petr Pivoňka.

[9] PIVOŇKA, P: *Optimalizace regulátorů.* VUT Brno, skriptum, 2005.

[10] ŠVANCARA, K.: *Lineární optimální regulátory*. UAMT VUT Brno, 2003.

[11] ANDERSEN, P. - JACOB, S.: *Optimal Control Lecture*. Aalborg University, 2007. Dostupné z: http://www.control.auc.dk/~pa/kurser/Optimal/optnote.pdf [citováno 2008-05-12]

[12] HAVLENA, V. - ŠTECHA, J.: *Moderní teorie řízení.* ČVUT, skriptum, 1999. Dostupné z: http://dce.felk.cvut.cz/stecha/mtr.pdf [citováno 2008-05-12]

[13] LORENC, V.: *LQ regulátor* (učební text). UAMT VUT Brno, 2007.

[14] ZAPLATÍLEK, K. - DOŇAR, B.: *MATLAB – tvorba uživatelských aplikací.* BEN – technická literatura (1. vydání), Praha, 2005. ISBN 80-7300-133-0.

[15] ZAPLATÍLEK, K. - DOŇAR, B.: *MATLAB – začínáme se signály.* BEN – technická literatura, Praha, 2006. ISBN 80-7300-200-0.

[16] VELEBA, V. - PIVOŇKA, P.: *Adaptive Controller with Identification Based on Neural Network for Systems with Rapid Sampling Rates*. WSEAS Transactions on Systems, 2005, vol. 4, issue 4, pp. 385-388. ISSN: 1109-2777.

[17] ZEZULKA, F. – BRADÁČ, Z. – FIEDLER, P. – KUČERA, P. – ŠTOHL. R.: *Programovatelné automaty*. VUT Brno, skriptum, 2003.

[18] PIVOŇKA, P. - SCHMIDT, M.: *Comparative Analysis of Discrete Derivative Implementations in PID Controllers*. Systems Theory and Applications, vol. 2, Řecko: WSEAS, 2007, s. 33-37. ISBN: 978-960-8457-90-4.

8. SEZNAM POUŽITÝCH ZKRATEK A SYMBOLŮ

w	žádaná hodnota
u	akční zásah
e	regulační odchylka
y	výstupní hodnota
$\hat{y}$	odhad výstupní hodnoty
v	měřitelný šum
e_s	náhodný šum (neměřitelný)
n	stochastický vliv
k	krok periody vzorkování
t	čas (spojitý)
z^{-1}	zpoždění o 1 krok
A	matice vnitřních vazeb systému (matice zpětných vazeb)
B	matice vazeb systému na vstup (vstupní matice)
C	matice vazeb výstupu na stav (výstupní matice)
D	matice přímých vazeb výstupu na vstup (výstupní matice)
x	stav systému
$F(s)$	spojitý přenos
$F(z)$	diskrétní přenos
T_s	perioda vzorkování
K	proporcionální konstanta / matice zpětných vazeb od stavů
T_I	integrační časová konstanta
T_D	derivační časová konstanta
N	filtrační konstanta
Δ	dopravní zpoždění
b	koeficient čitatele přenosu
a	koeficient jmenovatele přenosu

θ	vektor parametrů modelu
φ	vektor dat (měřených funkcí)
ε	chyba v kroku výpočtu
i	krok výpočtu
P	kovarianční matice
λ	koeficient zapomínání
I	jednotková matice
η	setrvačnost (momentum)
μ	konstanta (krok) učení (learning rate)
p	velikost bufferu (trénovací množiny)
S	penalizační matice konečného stavu systému
Q	penalizační matice odchylek stavů systému od nulových hodnot
R	penalizační matice energie pro řízení
Q_{se}	penalizační matice integrační akce
Q_p	penalizační matice stavů

ARX	Auto-Regressive with eXegenous variable
NN	neuronová síť (Neural Network)
TRM	trénovací množina
RLS	průběžná metoda nejmenších čtverců (Recursive Least Squares)
DNS	dopředná neuronová síť
BP	Back-propagation
LM	Levenberg-Marquardt
PID	proporcionálně integračně derivační regulátor
PSD	proporcionálně sumačně diferenční regulátor
S-PD	sumačně proporcionálně diferenční regulátor
LQ	Linear system, Quadratic cost
PLC	Programmable Logic Controller
B&R	Bernecker & Rainer
PVI	Process Visualization Interface

SEZNAM PŘÍLOH

Příloha 1	Skripty (MATLAB) pro identifikaci pomocí metody nejmenších čtverců (RLS)
Příloha 2	Skripty (MATLAB) pro identifikaci pomocí neuronové sítě (NN), metoda Back-propagation (BP)
Příloha 3	Skripty (MATLAB) pro identifikaci pomocí neuron. sítě (NN), metoda Levenberg-Marquardt (LM)

Příloha 1

Počáteční inicializace metody RLS

```
clear all;
close all;

global lambda;
global theta;
global fi;
global K;
global P;
global Ts;

Ts=0.5;
theta=[1 0 0 0 0 0]';
fi=[1 0 0 0 0 0]';
P=10^4*eye(6);
lambda=0.995;
```

Algoritmus metody RLS

```
function predikce=identRLS(y,u);

global lambda;
global theta;
global fi;
global K;
global P;
global Ts;
global k;
global TH;

K=P*fi*inv(lambda+(fi'*P*fi));
P=(1/lambda)*(P-(K*fi'*P));     %kovariancni matice
predikce=fi'*theta;
theta=theta+K*(y-predikce);

%aktualizace fi
fi=[u; fi(1:2); -y; fi(4:5)];
```

Příloha 2

Počáteční inicializace metody BP

```
clear all;
close all;

global w;
global w_predchozi;
global buffer;
global buffer_length;
global target;
global fi;
global iter_num;
global learningrate;
global momentum;
global Ts;
global predikce;

Ts=0.5;
fi=[1 0 0 0 0 0]';
w=[1 0 0 0 0 0]';
w_predchozi=[1 0 0 0 0 0]'

buffer_length=50;       %velikost trenovaci mnoziny
iter_num=10;            %pocet iteraci
buffer=zeros(6,buffer_length);
target=zeros(1,buffer_length);
learningrate=0.01;      %krok uceni
momentum=0.5
```

Algoritmus metody BP

```
function predikce=identNN_BP(y,u);

global w;
global w_predchozi;
global buffer;
global buffer_length;
global target;
global fi;
global iter_num;
global learningrate;
global momentum;
global Ts;
global predikce;

%aktualizace trenovaci mnoziny
buffer=[fi,buffer(:,1:buffer_length-1)];
target=[y,target(1,1:buffer_length-1)];
```

```
for j=1:1:iter_num
    for i=buffer_length:-1:1          %od posledniho sloupce
        predikce=buffer(:,i)'*w;      %vypocet vystupu NN
        e=target(i)-predikce;         %zpetne sireni chyby

        %aktualizace vah
        w=w+momentum*(w-w_predchozi)+(learningrate*e*buffer(:,i));
        w_predchozi=w;      %zaloha w(t-1)
    end
end

%aktualizace fi
fi=[u; fi(1:2); -y; fi(4:5)];
```

Příloha 3

Počáteční inicializace metody LM

```
clear all;
close all;

global fi;
global theta;
global learningrate;
global fi_mat;
global y_mat;
global Ts;
global predikce;
global buffer;
global iter_num;

Ts=0.5;
mat_length=50    %delka mnoziny trenovacich vzoru - volba: 50
iter_num=10      %volba: 10

fi=[1 0 0 0 0 0];
theta=[1 0 0 0 0 0]';

lambda=0.01       %krok uceni
fi_mat=zeros(6,mat_length);
y_mat=zeros(1,mat_length);
```

Algoritmus metody LM

```
function predikce=identNN_LM(y,u);

global fi;
global theta;
global learningrate;
global fi_mat;
global y_mat;
global Ts;
global predikce;
global buffer;
global iter_num;

%aktualizace
fi_mat=[fi',fi_mat(:,1:buffer-1)];  %fi_mat - vstupni matice
y_mat=[y,y_mat(:,1:buffer-1)];      %y_mat - vystupni vektor

for j=1:iter_num
    err=y_mat'-fi_mat'*theta;           %vektor chyb
    theta=theta-(fi_mat*fi_mat'+learningrate*eye(6,6))^(-1) ...
        *-fi_mat*err;   %aktualizace vah
end

%vypocet predikce
predikce=fi*theta;

%aktualizace fi
fi=[u, fi(1:2), -y, fi(4:5)];
```