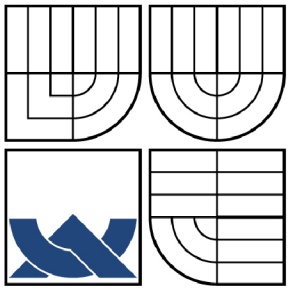
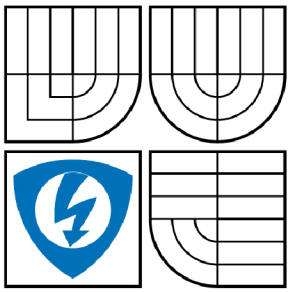


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

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

PROGRAM PRO TESTOVÁNÍ ADAPTIVNÍCH SYSTÉMŮ TYPU PSC

SOFTWARE TOOLS FOR TESTING OF ADAPTIVE CONTROL SYSTEMS PSC . (PARAMETER SCHEDULED SYSTEMS).

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — ALEŠ LEBEDA

AUTHOR

VEDOUCÍ PRÁCE — prof. Ing. PETR VAVŘÍN, DrSc.

SUPERVISOR

BRNO 2010

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

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student: Aleš Lebeda
Ročník: 3

ID: 109693
Akademický rok: 2009/10

NÁZEV TÉMATU:

Program pro testování adaptivních systémů typu PSC

POKYNY PRO VYPRACOVÁNÍ:

Vypracujte demonstrační program s metodikou použití pro instruktážní ukázky adaptivních systému typu PSC.

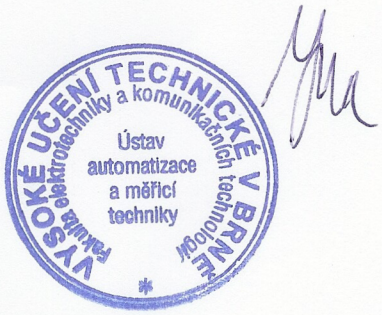
DOPORUČENÁ LITERATURA:

Vavřín: Teorie řízení 1
Manuály a popisy simulačního programu SIMULINK.

Termín zadání: ***Termín odevzdání:*** 31.5.2010

Vedoucí práce: prof. Ing. Petr Vavřín, DrSc.
Konzultanti bakalářské práce:

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



UPOZORNĚNÍ:

ABSTRAKT

Tato práce je zaměřena na problematiku adaptivních systémů typu PSC a metody jejich testování. Jejím obsahem je model adaptivního regulátoru typu PSC, který dovoluje uživateli navrhnout si vlastní adaptivní regulátor a poté i otestovat na uživatelem zadané soustavě s více stavy. Výstupem modelu je přechodová charakteristika, podle které si uživatel zhodnotí svůj návrh adaptivního regulátoru a poté má možnost svůj návrh změnit a opět otestovat.

KLÍČOVÁ SLOVA

software, simulace, adaptivní regulace, PSC, gain scheduling

ABSTRACT

This work is focused on problems of adaptive control systems PSC and their testing methods. Content of this work is a model of adaptive control system PSC that allows user to design his adaptive regulator and tests on user engaged system with more states. Output of the model is the step response which allows user to judge own adaptive control system. After that user can change his suggestion and test it again.

KEYWORDS

software, simulation, adaptive control, PSC, gain scheduling

LEBEDA, A. *Program pro testování adaptivních systémů typu PSC*. Bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2010. 50 s. Vedoucí práce: prof. Ing. Petr Vavřín, DrSc.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Program pro testování adaptivních systémů typu PSC“ jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské 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 autora)

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce prof. Ing. Petru Vavřínovi, DrSc. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Při řízení reálných regulačních systémů se můžeme setkat se systémy, které během provozu mohou změnit svoje vlastnosti. Tato změna způsobí, že použitý regulátor se stane neoptimální a řízený proces ztratí původní kvalitu. V určitých případech může dojít k nestabilitě systému, která může způsobit i různé druhy škod. Z tohoto důvodu je potřeba zareagovat na změnu tohoto systému a přizpůsobit parametry regulátoru. Regulátory, které se těmto změnám přizpůsobují, se nazývají adaptivní regulátory.

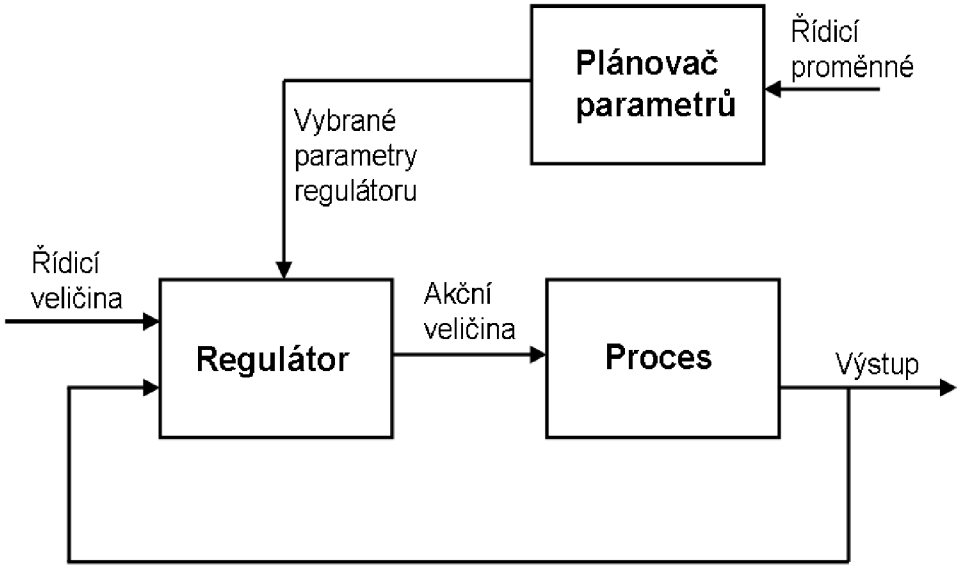
Nejznámější adaptivní regulátory jsou regulátory typu Model Reference Adaptive Control (MRAC), které se adaptují na základě referenčního modelu. Dále typu Self Tunning Regulators (STURE), které obsahují identifikační jednotku a jednotku pro návrh parametrů a poté se samočinně nastavují podle zvolené metody. Nebo regulátory typu Parametr Scheduling Control (PSC) v některých literaturách uváděných také jako ”Gain Scheduling”, které vyhodnocují v jakém pracovním bodě se nacházejí a podle toho přizpůsobují svoje parametry, které jsou pro jednotlivé pracovní body předem stanoveny. Tato práce se zabývá právě těmito posledně jmenovanými regulátory a možností jejich testování.

1 TEORETICKÝ ÚVOD

1.1 Parametr Scheduling Control

1.1.1 Princip

V některých případech můžeme najít pomocné proměnné, které dobře popisují změny v dynamice procesu. V takovémto případě je možné minimalizovat efekt změny parametrů procesu jednoduchou změnou parametrů regulátoru jako funkci pomocných proměnných viz. obr. 1.1. Hlavním problémem v návrhu PSC systému je výběr vhodných řídicích proměnných, pomocí kterých se budou měnit parametry regulátoru. K nalezení těchto proměnných, ale musíme dobře znát fyzikální vlastnosti daného procesu a vše, co s nimi souvisí.[1]



Obrázek 1.1: Blokový diagram PSC systému[2]

1.1.2 Zvolení vhodných řídicích proměnných

Ke zvolení vhodných řídicích proměnných nelze použít všeobecné pravidlo. Klíčovou otázkou je určení některé proměnné nebo proměnných, které můžou být použity jako řídicí proměnné pro změnu parametrů regulátoru. Pomocí těchto proměnných musíme být schopni jednoznačně rozpoznat jednotlivé stavy procesu.

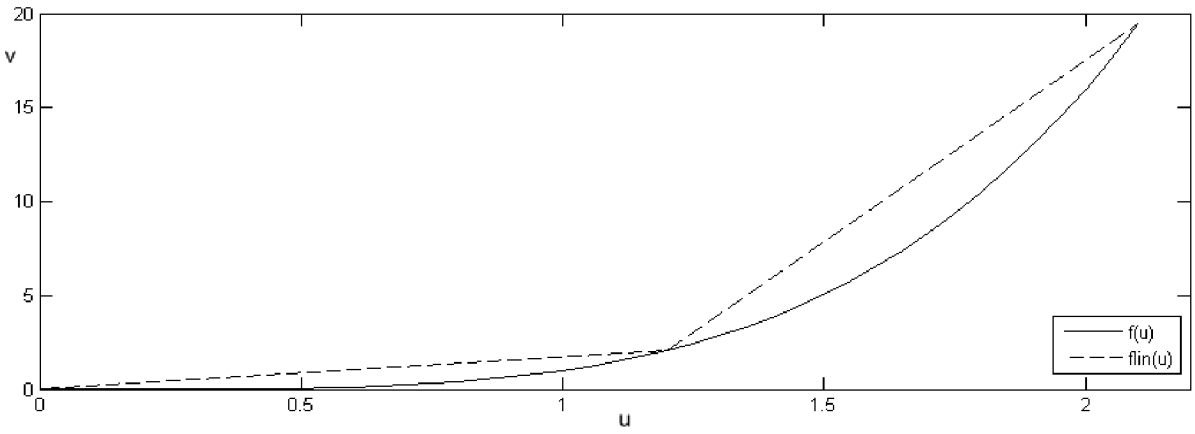
Jednou z možností je přímé měření některé pomocné proměnné. Touto proměnnou může být například v řízení letu letadla rychlost letu, výška nebo hmotnost zátěže. Při řízení pH lze využít právě hodnotu pH a například při řízení vstřikování v motoru u auta lze využít tok vzduchu a rychlost otáček motoru.

Další možností je rozdělení nelineárního procesu do více lineárních částí. Na obr. 1.2 je zachycena nelinearita nelineárního ventilu, která je rozdělena na dvě části. Nelinearita ventilu je popsána vztahem

$$v = f(u) = u^4 \qquad \text{pro } u \geq 0$$

kde v je výstupní veličina z nelinearity a u je vstupní veličina do nelinearity, která je v tomto případě akční zásah z regulátoru.

Za těchto okolností můžeme zvolit jako řídicí proměnnou, která bude řídit změnu parametrů regulátoru, akční zásah z regulátoru.[1, 2]



Obrázek 1.2: Nelineární charakteristika s aproximací na dvě části[1]

1.1.3 Shrnutí

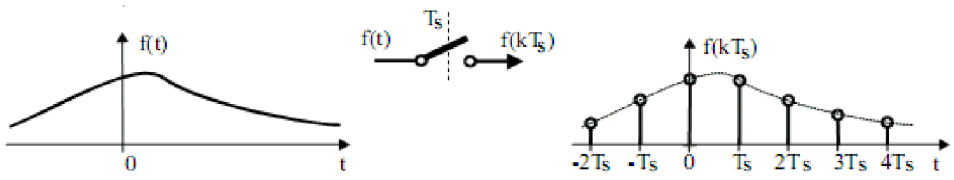
Výhodou PSC systému je, že parametry regulátoru mohou být změněny velice rychle po změně vlastností procesu. Jediným limitujícím faktorem je rychlost odezvy řídicích proměnných na změnu vlastností procesu. Naopak nevýhodou PSC systému je, že adaptace probíhá v otevřené smyčce bez jakéhokoli učení nebo inteligence, a tak tyto systémy nemají žádnou možnost kompenzovat nevhodně zvolené parametry.

Testovací software, který popisuje tato práce nepoužívá jako řízený proces nelineární soustavu, ale její linearizované části, kde každá tato část má svoji vlastní přenosovou funkci. Ke změně z jedné přenosové funkce do druhé dochází v předem nastaveném okamžiku, který je volitelný uživatelem tohoto softwaru. Jako řídicí proměnná je zde výstup z identifikace, která nabývá hodnot jedna až čtyři podle toho, v jaké linearizované části se systém zrovna nachází.

1.2 Diskrétní systémy

1.2.1 Vzorkovací teorém

V našem okolí se nacházejí většinou spojité signály, které ale chceme v některých případech zpracovávat pomocí počítače. Zpracovávat přímo spojitý signál počítačem není možné, protože počítače pracují s diskrétním časem. Proto se musí tento spojitý signál převést na posloupnost čísel, reprezentujících původní spojitý signál. Tento převod se provádí pomocí vzorkování. Vzorkování znamená odebrání vzorku ze spojitého signálu v pravidelných časových okamžicích. Tyto pravidelné časové okamžiky se značí T_s. Průběh vzorkování je zobrazen na obr. 1.3.



Obrázek 1.3: Vzorkování signálu[8]

Spínač spíná v pravidelných časových okamžicích kT_s pro $k = \ldots,-2,-1,0,1,2,\ldots$ na velmi krátkou dobu a na jeho výstupu se získá pro každý časový okamžik hodnota $f(kT_s)$. Tato hodnota se může uložit do paměti počítače a následně zpracovávat podle potřeby. Nabízí se ale otázka, za jakých podmínek lze zpětně ze získaných vzorků zrekonstruovat původní spojitý signál. Tuto otázku řeší vzorkovací teorém vztah 1.1 nebo také Shannon-Kotelníkův teorém pojmenovaný po americkém matematikovi Claudu Elwoodu

Shannonovi (1916-2001) a po ruském matematikovi Vladimiru Alexandroviči Kotelnikovi (1908-2005). Vzorkovací teorém je vyjádřen vztahem

$$\omega_T = \frac{2\pi}{T_s} \geq 2\omega_{max} \tag{1.1}$$

kde ω_T je vzorkovací frekvence, T_s je perioda vzorkování a ω_{max} je maximální frekvence vyskytující se v signálu.

Tato podmínka říká, že pokud je spojitý signál vzorkován alespoň dvakrát rychleji než je nejvyšší frekvence vyskytující se v tomto signálu, potom nedochází při vzorkování ke ztrátě informace.[3, 8]

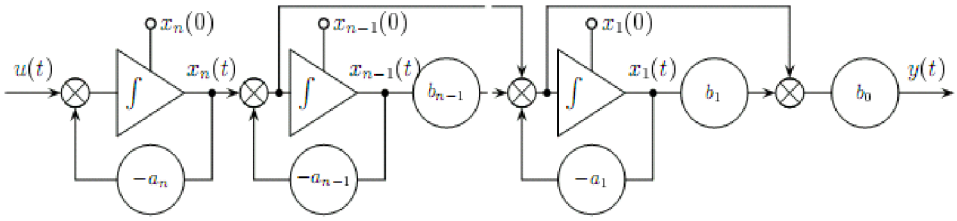
1.2.2 Použití diskrétní soustavy

V první verzi programu byla použita spojitá verze soustavy, ale ta se ukázala být nevhodná pro potřeby simulace. V programu Simulink lze použít v simulačním schématu dva základní bloky, které zastupují přenosovou funkci soustavy. Jsou jimi Transfer function a LTI system. Oba dva bloky ale nelze použít, protože nedovolují změnit svoje parametry během simulace, což je důležitá podmínka pro simulaci soustavy s parametry, které lze během simulace měnit. Další možností je použití stavového diagramu. Stavový diagram propojuje základní stavební prvky tak, aby popisovaly chování nějakého systému. Základní stavební prvky jsou integrátor, sumátor a proporcionální člen. První použitou metodou pro stavový diagram bylo sériové programování. Použití sériového programování je vhodné, pokud je přenosová funkce ve tvaru součinu kořenových činitelů.

$$F(p) = \frac{b_0(p+b_1)(p+b_2)...(p+b_m)}{(p+a_1)(p+a_2)...(p+a_n)} \tag{1.2}$$

Stavový diagram, který odpovídá přenosu 1.2, je zobrazen na obrázku 1.4. Tvoří jej kaskádní spojení elementárních bloků, které odpovídají jednotlivým pólům a nulám přenosu.

Při použití sériového programování už lze měnit jednotlivé parametry během simulace. Jedním takovým způsobem je použití s-funkce místo proporcionálního zesílení ve zpětné vazbě, která se chová jako proporcionální zesílení, ale v určité chvíli nebo za určitých podmínek lze velikost tohoto zesílení změnit. Problém ale nastává, když přenos obsahuje ve jmenovateli komplexní kořeny. Potom se musí použít přímé programování.

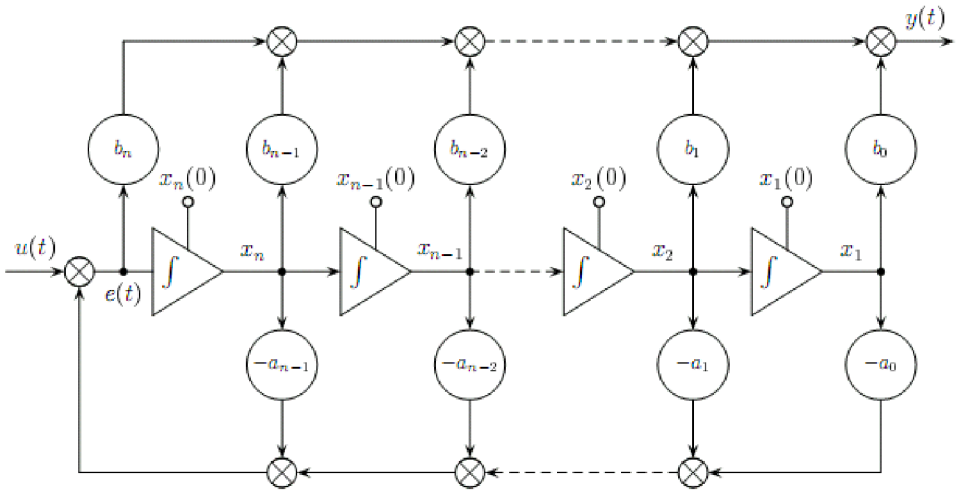


Obrázek 1.4: Stavový diagram sériového programování[6]

Přímé programování je vhodné, jestliže přenosová funkce je ve tvaru dvou polynomů. Pomocí přímého programování již lze pracovat i s komplexními kořeny.

$$F(p) = \frac{b_m p^m + b_{m-1} p^{m-1} + \ldots + b_1 p + b_0}{a_n p^n + a_{n-1} p^{n-1} + \ldots + a_1 p + a_0} \tag{1.3}$$

V rovnici 1.3 musí být řád čitatele menší nebo roven řádu jmenovatele $m \leq n$. Stavový diagram, který odpovídá tomuto přenosu 1.3 je na obrázku 1.5. Hodnota koeficientu a_n je rovna jedné kvůli realizaci. Použití přímého programování se změnou parametrů během simulace se ukázalo být nevhodné, protože při změně parametrů se výsledný systém chová nepřirozeně. Toto chování je způsobeno okamžitou změnou parametru b_0, na kterou nedokáže integrátor na výstupu stavového diagramu přímého programování dostatečně rychle zareagovat.

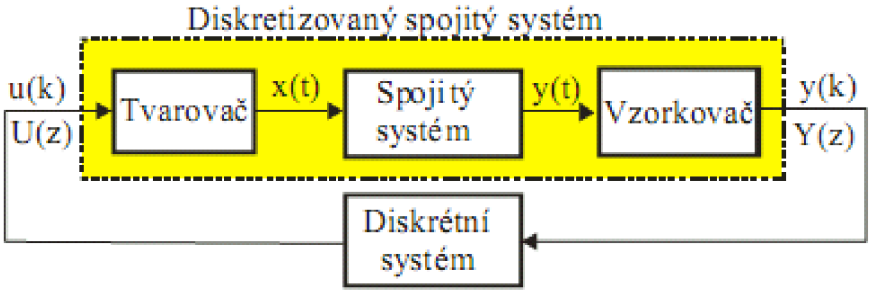


Obrázek 1.5: Stavový diagram přímého programování[6]

Z výše uvedených důvodů je při simulaci použit diskrétní model soustavy, který je získán pomocí diskretizace spojitého přenosu, kterou zadá uživatel do vyznačených polí. Více je uvedeno v části 2.3.1.[6]

1.2.3 Diskretizace

Diskretizovaný spojitý systém se získá, pokud na výstup spojitého systému se zařadí vzorkovač a tím se získá ze spojitého signálu diskrétní signál. Na vstup spojitého systému se zařadí tvarovač (rekonstruktor signálu) a tím se získá z diskrétního signálu signál se spojitým časem. Tato situace je zachycena na obr. 1.6.



Obrázek 1.6: Diskretizace spojitého systému[8]

Na obrázku je vidět, že do diskretizovaného spojitého systému vstupuje diskrétní signál $u(k)$ a vystupuje z něj diskrétní signál $y(k)$. Tyto signály lze označit jako Z-obrazy $U(z)$, $Y(z)$. Z toho je zřejmé, že lze zavést pojem ekvivalentní Z-přenos diskretizovaného spojitého systému jako

$$F_e(z) = \frac{Y(z)}{U(z)}. \tag{1.4}$$

Předpokládejme, že spojitý systém je stabilní, řádu n a má operátorový přenos

$$F(p) = \frac{B_m(p)}{A_n(p)}. \tag{1.5}$$

Z daného přenosu se vypočte časový průběh přechodové charakteristiky podle vztahu 1.6 a spojitý signál se ovzorkuje s periodou T_s a tím se získá posloupnost $y(k)$.

$$h(t) = \mathscr{L}^{-1}\{H(p)\} = \mathscr{L}^{-1}\left\{\frac{1}{p}F(p)\right\} \tag{1.6}$$

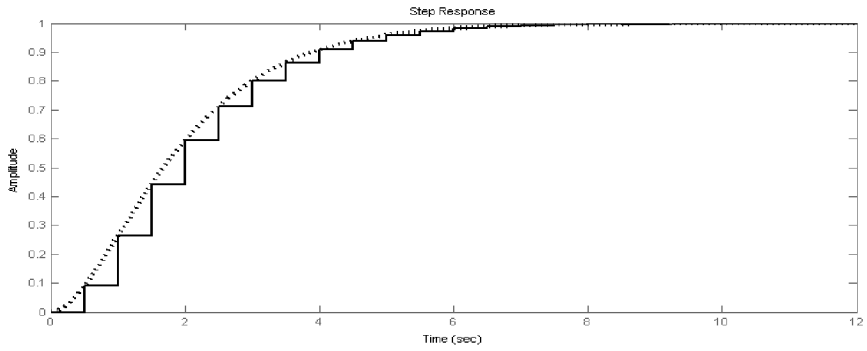
Nakonec se najde Z-obraz této posloupnosti $Y(z)$ dle vztahu 1.7 a pro ekvivalentní Z-přenos platí vztah 1.8.

$$Y(z) = \mathscr{Z}\{y(kT)\} \tag{1.7}$$

$$F_e(z) = (1 - z^{-1})Y(z) = \frac{z-1}{z}Y(z) \tag{1.8}$$

Uvedený postup platí jen pro tvarovač nultého řádu (Č/A převodník) s přenosem dle vztahu 1.9. Ukázka signálu rekonstruovaného pomocí tvarovače nultého řádu je na obr. 1.7, kde čárkovaně je původní signál a plnou čarou signál rekonstruovaný pomocí tvarovače nultého řádu. Pro jiný typ tvarovače by se musel použít jiný postup.

$$F_{Tv}(p) = \frac{1 - e^{T_s p}}{p} \tag{1.9}$$



Obrázek 1.7: Porovnání původního signálu s rekonstruovaným signálem

V programu, kterým se zabývá tato práce je soustava diskretizována pomocí funkce *c2d*. Její použití je ukázáno na příkladu 1.1.

Příklad 1.1: Ukázka použití funkce c2d

```
F1=c2d(tf(num1,den1),Tss);
```

V příkladě 1.1 je vidět, že ve funkci *c2d* jsou použity dva vstupní parametry. Prvním vstupním parametrem je přenos soustavy, kterou chceme diskretizovat. V tomto příkladě je přenos soustavy vytvořen funkcí *tf*, která má dva vstupní parametry, kterými jsou koeficienty polynomu čitatele *num1* a koeficienty polynomu jmenovatele *den1*. Druhým vstupním parametrem je velikost vzorkovací periody, jakou má být soustava vzorkována. Lze použít i třetí parametr, kterým je zvolení typu tvarovače pro rekonstrukci signálu. Ve

výchozím nastavení je ale zvolen tvarovač nultého řádu, a proto není nutné tento parametr používat.

Po získání ekvivalentního Z-přenosu pomocí diskretizace můžeme vypočítat diferenční rovnici, která je použita pro výpočet aktuální hodnoty soustavy v simulačním modelu. Předpokládejme, že máme diskrétní přenos ve tvaru

$$F(z) = \frac{Q_m(z)}{P_n(z)} = \frac{Y(z)}{U(z)} = \frac{b_n z^0 + b_{n-1} z^{-1} + \ldots + b_0 z^{-n}}{a_n z^0 + a_{n-1} z^{-1} + \ldots + a_0 z^{-n}}. \tag{1.10}$$

Rovnice ze vztahu 1.10 se upraví do tvaru

$$Q_m(z)U(z) = P_n(z)Y(z) \tag{1.11}$$

a poté se na obě strany rovnice 1.11 aplikuje zpětná Z-transformace

$$\mathscr{Z}^{-1}\{Q_m(z)U(z)\} = \mathscr{Z}^{-1}\{P_n(z)Y(z)\}. \tag{1.12}$$

Z rovnice 1.12 se získá diferenční rovnice ve tvaru

$$a_n y(k) + a_{n-1} y(k-1) + \ldots + a_0 yk - n = b_n u(k) + b_{n-1} u(k-1) + \ldots + b_0 uk - n \tag{1.13}$$

a ta je následně upravena pro výpočet hodnoty *y(k)* do tvaru

$$y(k) = \frac{b_n u(k) + b_{n-1} u(k-1) + \ldots + b_0 uk - n - a_{n-1} y(k-1) - \ldots - a_0 yk - n}{a_n}. \tag{1.14}$$

Při použití funkce *c2d* je diskretizovaný přenos upraven do tvaru, ve kterém je koeficient $a_n = 1$.[6, 8]

1.2.4 PSD regulátor

Většina dnes vyráběných průmyslových regulátorů je diskrétního typu. PSD (Proporcionálně Sumačně Diferenční) regulátor je diskrétní verzí PID (Proporcionálně Integračně Derivačního) regulátoru. Protože PSD regulátor vychází z PID regulátoru, tak je snaha se mu co nejvíce přiblížit. Podmínkou nutnou, ale nikoliv postačující, je dodržení vzorkovacího teorému, který je popsán v části Vzorkovací teorém 1.2.1. Další podmínkou je, aby byly rovněž spojitým filtrem dostatečně potlačeny všechny rušivé signály vyšší, než odpovídající frekvence vzorkování. Tedy perioda vzorkování byla dostatečně krátká vzhledem k dynamice reálného nebo modelovaného systému a derivační složka PSD regulátoru byla dostatečně vyfiltrována.

Základní tvar spojitého PID regulátoru je uveden vztahem

$$u(t) = K\left(e(t) + \frac{1}{T_I}\int_0^t e(t)\,dt + T_D\frac{de(t)}{dt}\right) \tag{1.15}$$

kde K je zesílení PID regulátoru, T_I je integrační konstanta regulátoru, T_D je derivační konstanta regulátoru, $e(t)$ je regulační odchylka, která je rovna rozdílu požadované veličiny $w(t)$ a její skutečné hodnoty $y(t)$; $e(t) = w(t) - y(t)$, a $u(t)$ je velikost akčního zásahu z regulátoru

Při použití Laplaceovy transformace na vztah 1.15 se získá Laplaceův obraz

$$U(p) = K\left(E(p) + \frac{1}{T_I p}E(p) + pT_D E(p)\right) \tag{1.16}$$

a po úpravě vztahu 1.16 na přenos regulátoru se získá tvar

$$F(p) = \frac{U(p)}{E(p)} = K\left(1 + \frac{1}{T_I p} + pT_D\right). \tag{1.17}$$

Derivační část přenosu regulátoru ve vztahu 1.17 není ale fyzikálně realizovatelná. Z tohoto důvodu se zavádí do derivační části časová konstanta ε, kde $\varepsilon = \frac{T_D}{N}$. Hodnota N omezuje zesílení na vyšších frekvencích a její velikost se volí v rozmezí 3 až 20. Pro hodnotu $N = 3$ je signál nejvíce filtrován.

$$F_R(p) = K\left(1 + \frac{1}{T_I p} + \frac{T_D p}{\varepsilon p + 1}\right) = K\left(1 + \frac{1}{T_I p} + \frac{T_D p}{\frac{T_D}{N}p + 1}\right) \tag{1.18}$$

Aby bylo možné realizovat PSD regulátor jako s-funkci v simulačním prostředí Simulink programu MATLAB, je potřeba převést rovnici 1.18 do diskrétního tvaru pomocí diskretizace (část Diskretizace 1.2.3) a následně pomocí zpětné Z-transformace převést do diferenčního tvaru.

Nejdříve je potřeba diskretizovat proporcionální část z přenosu 1.18.

$y_P(t) = \mathscr{L}^{-1}\left\{\frac{1}{p}K\right\} = K\sigma(t)$

$y_P(k) = y_P(t)\|_{t=kT_s} = y_P(kT_s) = K\sigma(k)$

$Y_P(z) = \mathscr{Z}\{y_P(kT_s)\} = \mathscr{Z}\{K\sigma(k)\} = \frac{K}{1-z^{-1}}$

$$F_P(z) = (1-z^{-1})Y_P(z) = (1-z^{-1})\frac{K}{1-z^{-1}} = K \tag{1.19}$$

Ve výsledku diskretizace proporcionální části je vidět, že zesílení K zesiluje K-krát a to jak ve spojitém, tak i diskrétním přenosu.

Poté je potřeba diskretizovat integrační část z přenosu 1.18.

$$y_I(t) = \mathscr{L}^{-1}\left\{\frac{1}{p}\frac{1}{p}\frac{K}{T_I}\right\} = \mathscr{L}^{-1}\left\{\frac{1}{p^2}\frac{K}{T_I}\right\} = t\frac{K}{T_I}$$

$$y_I(k) = y_I(t)\|_{t=kT_s} = y_I(kT_s) = kT_s\frac{K}{T_I}$$

$$Y_I(z) = \mathscr{Z}\{y_I(kT_s)\} = \mathscr{Z}\left\{\frac{KT_s}{T_I}\right\} = \frac{KT_s}{T_I}\mathscr{Z}\{k\} = \frac{KT_s}{T_I}\frac{z}{(z-1)^2}$$

$$F_S(z) = \frac{z-1}{z}Y(z) = \frac{z-1}{z}\frac{KT_s}{T_I}\frac{z}{(z-1)^2} = \frac{KT_s}{T_I}\frac{1}{z-1} = \frac{KT_s}{T_I}\frac{z^{-1}}{1-z^{-1}} \tag{1.20}$$

Nakonec se diskretizuje derivační část z přenosu 1.18.

$$y_D(t) = \mathscr{L}^{-1}\left\{\frac{1}{p}\frac{T_D p}{\frac{T_D}{N}p+1}\right\} = \mathscr{L}^{-1}\left\{\frac{N}{p+\frac{N}{T_D}}\right\} = KNe^{\frac{-Nt}{T_D}}$$

$$y_D(k) = y_D(t)\|_{t=kT_s} = KNe^{\frac{-NkT_s}{T_D}}$$

$$Y_D(z) = \mathscr{Z}\{y_D(kT_s)\} = \mathscr{Z}\left\{KNe^{\frac{-NkT_s}{T_D}}\right\} = KN\frac{1}{1-z^{-1}e^{\frac{-NT_s}{T_D}}}$$

$$F_D(z) = (1-z^{-1})Y_D(z) = KN\frac{1-z^{-1}}{1-e^{\frac{-NT_s}{T_D}}z^{-1}} \tag{1.21}$$

Sečtením Z-obrazů proporcionální (vztah 1.19), sumační (vztah 1.20) a diferenční (vztah 1.21) části se získá výsledný přenos PSD regulátoru s filtrací derivační složky.

$$F_R(z) = F_P(z) + F_I(z) + F_D(z) \tag{1.22}$$

$$F_R(z^{-1}) = \frac{U(z^{-1})}{E(z^{-1})} = K\left(1 + \frac{T_s}{T_I}\frac{z^{-1}}{1-z^{-1}} + N\frac{1-z^{-1}}{1-e^{\frac{-NT_s}{T_D}}z^{-1}}\right) \tag{1.23}$$

Přenos PSD regulátoru s filtrací derivační složky 1.23 se nakonec převede do diferenčního tvaru a tím je možné ho implementovat do programovacího jazyka C a použít například v průmyslovém automatu nebo v s-funkci pro simulování v programu MATLAB.

Výpočet diferenční rovnice proporcionální složky z rovnice 1.23.

$$F_P(z) = K = \frac{U_P(z)}{E(z)}$$

$$U_P(z) = KE(z)$$

$$\mathscr{Z}^{-1}\{U_P(z)\} = K\mathscr{Z}^{-1}\{E(z)\}$$

$$u_P(k) = Ke(k) \tag{1.24}$$

Výpočet diferenční rovnice sumační složky z rovnice 1.23.

$$F_I(z) = \frac{KT_s}{T_I}\frac{z^{-1}}{1-z^{-1}} = \frac{U_I(z)}{E(z)}$$

$$U_I(z) - z^{-1}U_I(z) = \frac{KT_s}{T_I}z^{-1}E(z)$$

$$\mathscr{Z}^{-1}\{U_I(z) - z^{-1}U_I(z)\} = \mathscr{Z}^{-1}\left\{\frac{KT_s}{T_I}z^{-1}E(z)\right\}$$

$$u_I(k) = \frac{KT_s}{T_I}e(k-1) + u_I(k-1) \tag{1.25}$$

Výpočet diferenční rovnice diferenční složky z rovnice 1.23.

$$F_D(z) = KN\frac{1-z^{-1}}{1-e^{\frac{-NT_s}{T_D}}z^{-1}} = \frac{U_D(z)}{E(z)}$$

$$U_D(z) - e^{\frac{-NT_s}{T_D}}z^{-1}U_D(z) = KNE(z) - KNz^{-1}E(z)$$

$$\mathscr{Z}^{-1}\{U_D(z) - e^{\frac{-NT_s}{T_D}}z^{-1}U_D(z)\} = \mathscr{Z}^{-1}\left\{KNE(z) - KNz^{-1}E(z)\right\}$$

$$u_D(k) = KN(e(k) - e(k-1)) + e^{\frac{-NT_s}{T_D}}u_D(k-1) \tag{1.26}$$

Nakonec zbývá jen sečíst dohromady proporcionální (vztah 1.24), sumační (vztah 1.25) a diferenční (vztah 1.26) složky vypočtené při zpětné Z-transformaci.

$$u(k) = u_P(k) + u_I(k) + u_D(k) \tag{1.27}$$

Ze vztahu 1.27 vychází všechny použité typy regulátorů v programu.[1, 3, 6, 8]

2 PSC TESTOVACÍ SW

2.1 Struktura programu

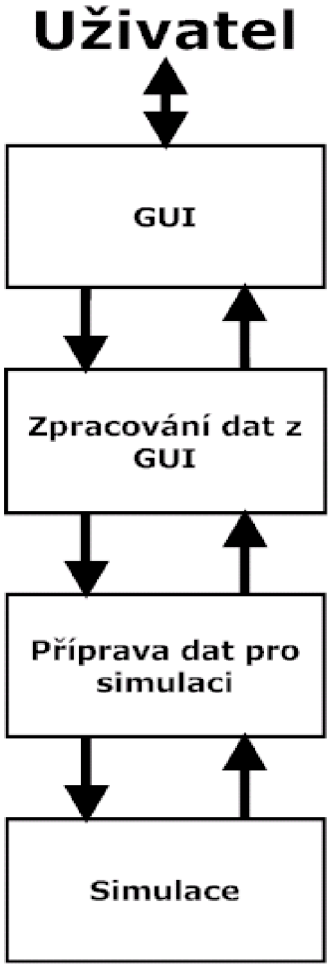
Program PSC testovací SW je vytvořen v programu MATLAB a využívá jeho nadstavby Simulink pro simulaci přechodového děje. Samotný program se skládá z devíti souborů. Struktura programu je znázorněna na obrázku 2.1. Uživatel zadá požadovaná data do GUI (grafické uživatelské rozhraní – Graphical user interface) a spustí simulaci. Data z grafického uživatelského rozhraní jsou načtena do vhodného formátu. Následně jsou zpracována a podle získaných informací je nastavena simulace, která je následně spuštěna, pokud nenastala nějaká chyba. Data ze simulace jsou uložena do paměti a upravena pro použití v grafickém uživatelském rozhraní. Nakonec jsou získaná data ze simulace zobrazena v grafickém uživatelském rozhraní, kde uživatel vidí výsledek svého nastavení, které je v případě tohoto programu přechodová funkce.

Rozdělení souborů do jednotlivých struktur z obr. 2.1:

- **GUI** - *pscsys.fig* - rozložení jednotlivých prvků v grafickém uživatelském prostředí
- **Zpracování dat z GUI** - *pscsys.m* - vlastnosti jednotlivých prvků grafického uživatelského prostředí a nastavení funkcí, které se zavolají při jejich použití (stisknutí, zaškrtnutí, atd.)
- **Příprava dat pro simulaci** - *PSC_system_config.m* - hlavní soubor zajišťující zpracování a kontrolu dat získaných v předchozím kroku a následné spuštění simulace
- **Simulace** - *PSC_system.mdl, sfunident.m, sfunPSD.m, sfunsys.m, sfunw.m* - simulace systému

2.2 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní je uživatelské rozhraní, které může uživatel ovládat pomocí grafických prvků. Jeho cílem je uživateli co nejvíce usnadnit práci a pomoci se, co nejrychleji orientovat v daném programu. Uživatel nemusí znát žádné složité funkce, které

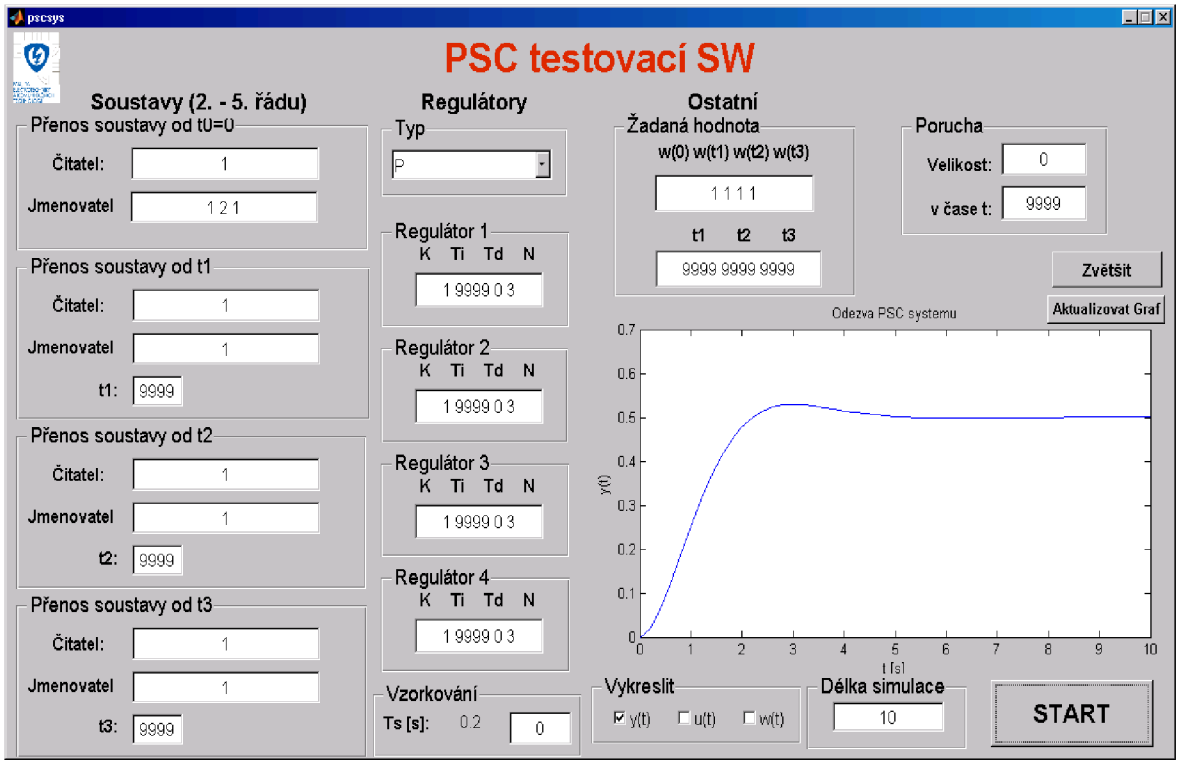


Obrázek 2.1: Struktura programu PSC testovací SW

jsou volané pomocí grafických prvků, ale stačí aby například klik pomocí myši na tlačítko a funkce se vykoná.

V programu MATLAB lze použít pro tvorbu grafického uživatelského rozhraní nástroj na tvorbu těchto rozhraní, který se jmenuje GUIDE (Graphical User Interface Development Environment). Tento nástroj se spouští pomocí konzole v programu MATLAB, do které se zadá příkaz *guide* a potvrdí. Tento nástroj dovoluje programátorovi vytvářet vlastní vzhled grafického uživatelského prostředí, posouvat a zarovnávat jednotlivé prvky. Poté co jsou prvky na místě, programátor může měnit jejich vlastnosti jako název, barvu, velikost prvku nebo velikost textu. Při uložení grafického uživatelského prostředí se automaticky vytvoří hlavní kostry funkcí pro každý prvek použitý v grafickém uživatelském rozhraní.

Program PSC testovací Sw obsahuje grafické uživatelské rozhraní pro usnadnění a zpřehlednění zadávaných dat pro simulaci adaptivního regulátoru. Toto grafické uživatelské rozhraní je zobrazeno na obr. 2.2.



Obrázek 2.2: Grafické uživatelské rozhraní programu PSC testovací SW

2.2.1 Tlačítko

Jedním ze základních ovládacích prvků je tlačítko, které po stisknutí vykoná předem nastavenou akci. Tlačítko je zobrazeno na obr. 2.3. Tlačítko si programátor může upravit podle svojí potřeby. Může u něj měnit velikost, barvu a popisek.

Obrázek 2.3: Tlačítko

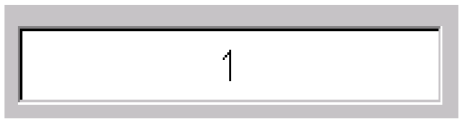
Při vytvoření tlačítka se po uložení vygeneruje funkce, která se spustí při stisku tlačítka. Do této funkce si uživatel napíše vlastní kód, který je při stisku tlačítka, které zavolá tuto funkci, vykonán. Vygenerovaná funkce je ukázána na příkladu 2.1. Název prvku tlačítka v příkladu 2.1 je *start_button* a funkce obsahuje tři vstupní parametry. Prvním parametrem je objekt *hObject*, který je objektem příslušné funkce a chová se jako lokální proměnná. Druhý parametr je objekt *evendata*, který je rezervovaný pro budoucí verze programu MATLAB. Třetí parametr je objekt *handles*, který se chová jako globální proměnná. K objektu *handles* mají přístup všechny funkce.[9]

Příklad 2.1: Funkce start_button_Callback

```
function start_button_Callback (hObject, eventdata, handles)
% hObject    handle to start_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

2.2.2 Edit Text

Dalším používaným prvkem je Edit Text, který dovoluje uživateli do něj zapsat libovolný řetězec, který se následně může načíst a zpracovat. Příklad prvku Edit Text je na obr. 2.4. Programátor si může zvolit podle potřeb velikost prvku a vlastnosti písma.



Obrázek 2.4: Edit Text

Při vytvoření prvku Edit Text se po uložení vygenerují dvě funkce. První funkce čeká na zavolání z jiné části programu a poté provede obsažený kód. Ukázka této funkce je v příkladě 2.2. Druhá funkce se automaticky vykoná během vytvoření prvku Edit Text a do této části není potřeba přidávat žádný kód. Ukázka této funkce je v příkladě 2.3. Název prvku Edit Text je *simtime*.

Příklad 2.2: Funkce simtime_Callback

```
function simtime_Callback(hObject, eventdata, handles)
% hObject    handle to simtime (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of simtime as text
%        str2double(get(hObject,'String')) returns contents of simtime as a double
input = str2num(get(hObject,'String'));
if (isempty(input))
    handles.prepinac=99; errordlg('Spatne_zadan_cas_simulace!','Simulace_error');
end
guidata(hObject, handles);
```

K této funkci je přidán zdrojový kód, který převede řetězec zadaný do pole Edit Textu na číslo a uloží do lokální proměnné *input* a pokud řetězec není číslo, tak do lokální proměnné *input* neuloží nic a ta zůstane prázdná. Následuje příkaz *if*, který vyhodnocuje jestli je lokální proměnná *input* prázdná. Pokud je lokální proměnná *input* prázdná, je nastavena globální proměnná *handles.prepinac* na hodnotu 99 a zobrazí se chybové hlášení. Poslední příkaz aktualizuje objekty *hObject* a *handles*.[9]

Příklad 2.3: Funkce simtime_CreateFcn

```
function simtime_CreateFcn(hObject, eventdata, handles)
% hObject    handle to simtime (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

2.2.3 Check Box

Prvek Check Box má dva stavy, zaškrtnut nebo nezaškrtnut. Používá se pro ovládání věcí, které mají dva stavy ON nebo OFF. Tento prvek je zobrazen na obrázku 2.5.

Obrázek 2.5: Check Box

Při vytvoření prvku Check Box se po uložení vygeneruje funkce. Tato funkce čeká na zavolání z jiné části programu a poté provede obsažený kód. Ukázka této funkce je v příkladě 2.4. Název prvku Check Box je *scopey*.

Příklad 2.4: Funkce scopey_Callback

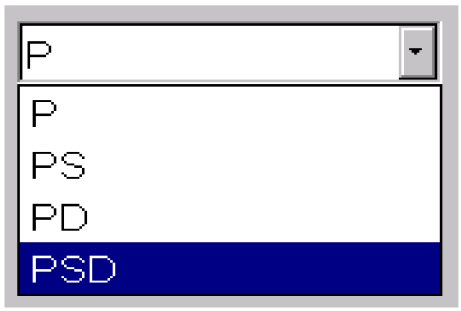
```
function scopey_Callback(hObject, eventdata, handles)
% hObject    handle to scopey (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of scopey
checkboxStatus = get(handles.scopey,'Value');
if (checkboxStatus)
    handles.CheckBox(1)=1;
else
    handles.CheckBox(1)=0;
end
guidata(hObject, handles);
```

K této funkci je přidán zdrojový kód, který uloží do lokální proměnné *checkboxStatus* hodnotu nula nebo jedna. Pokud tento Check Box je zaškrtnut, uloží se do lokální proměnné *checkboxStatus* hodnota jedna. Pokud není zaškrtnut, uloží se do ní hodnota nula. Následuje příkaz *if*, který podle hodnoty v lokální proměnné *checkboxStatus* nastaví hodnotu prvního prvku v globální proměnné *handles.CheckBox*. Poslední příkaz aktualizuje objekty *hObject* a *handles*.[9]

2.2.4 Pop-up Menu

Prvek Pop-up Menu se používá v případě, že programátor chce omezit volbu na určité možnosti. Při kliknutí na Pop-up Menu se zobrazí nabídka všech možností, ze kterých si uživatel může vybrat. Tento prvek je zobrazen na obrázku 2.6.



Obrázek 2.6: Pop-up Menu

Při vytvoření prvku Pop-up Menu se po uložení vygenerují dvě funkce. První funkce čeká na zavolání z jiné části programu a poté provede obsažený kód. Ukázka této funkce je v příkladě 2.5. Druhá funkce se automaticky vykoná během vytvoření prvku Pop-up

Menu a do této části není potřeba přidávat žádný kód. Ukázka této funkce je v příkladě 2.6. Název prvku Pop-up Menu je *regulator_type*.

Příklad 2.5: Funkce regulator_type_Callback

```
function regulator_type_Callback(hObject, eventdata, handles)
% hObject    handle to regulator_type (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns regulator_type contents as cell array
%        contents{get(hObject,'Value')} returns selected item from regulator_type
str = get(hObject, 'String');
val = get(hObject,'Value');
handles.type=str{val};
guidata(hObject,handles)
```

K této funkci je přidán zdrojový kód, který uloží do lokální proměnné *str* pole řetězců zadaných v prvku Pop-up Menu. Následující příkaz uloží do lokální proměnné *val* index vybraného prvku. Do globální proměnné *handles.type* je potom uložen vybraný řetězec. Poslední příkaz aktualizuje objekty *hObject* a *handles*.[9]

Příklad 2.6: Funkce regulator_type_CreateFcn

```
function regulator_type_CreateFcn(hObject, eventdata, handles)
% hObject    handle to regulator_type (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

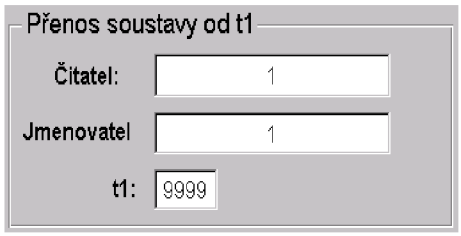
2.3 Použití grafického uživatelského rozhraní

Tento program není samostatně spustitelný a ke svému běhu potřebuje právě program MATLAB s jeho nadstavbou Simulink. Pro spuštění programu je třeba mít nastavený aktuální adresář na adresář obsahující soubory tohoto programu a do konzole programu MATLAB se zadá příkaz **pscsys**. Poté se spustí grafické uživatelské prostředí programu PSC testovací SW. Grafické uživatelské rozhraní je uspořádáno pro snadnou a intuitivní orientaci v programu. V levé části rozhraní se nastavuje řízená soustava a její další stavy. Vedle nastavení soustavy se nastaví typ regulátoru a příslušné parametry pro každý stav soustavy. V levé dolní části lze změnit velikost vzorkování, kdyby zvolená velikost vzorkování byla nevhodná, protože perioda vzorkování se volí jako jedna pětina reálné části

největší časové konstanty výchozího stavu soustavy a toto nastavení může být nevhodné pro ostatní stavy soustavy. V pravé části rozhraní je nastavení žádané hodnoty $w(t)$ v čase a nastavení velikosti poruchy, která vstupuje do soustavy, a času, kdy tato porucha začne působit. V pravé dolní části je vidět výsledný přechodový děj, nastavení délky simulace a tlačítko pro spuštění simulace.

2.3.1 Nastavení soustavy

Soustava je v programu rozdělena na čitatel a jmenovatel (obr. 2.7). Každý člen se zadává zvlášť ve tvaru koeficientů příslušného polynomu. Ukázka přenosu je v rovnici 1.3. Řád čitatele musí být menší než řád jmenovatele, řád jmenovatele nemůže být menší než dva a nejvyšší možný řád jmenovatele je pět. Do pole čitatele se zadávají koeficienty b_m až b_0 oddělené mezerou a do pole jmenovatele se zadávají koeficienty a_n až a_0 také oddělené mezerou. Je-li v poli pro jmenovatel nebo pro čitatel zadán jiný znak než číslo, tak je při spuštění simulace zobrazena chyba, která má v popisu označen špatně zadaný řetězec. Soustava přechází do dalšího stavu v čase, který zadá uživatel. Pořadí jednotlivých stavů soustavy je stav1 → stav2 → stav3 → stav4. Toto pořadí musí být dodrženo.

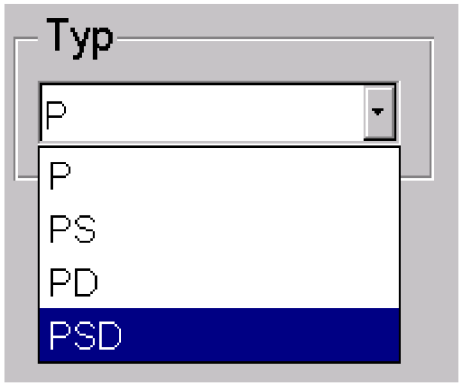


Obrázek 2.7: Nastavení druhého stavu soustavy

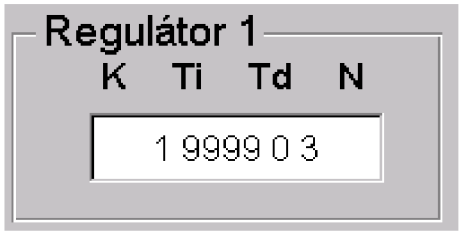
2.3.2 Nastavení regulátoru

Pro každý stav soustavy je možné nastavit jiné parametry regulátoru. Program umožňuje zvolit regulátory typu P, PS, PD a PSD. Okno s výběrem typu regulátoru je na obrázku 2.8. Všechny parametry regulátoru se zadávají do jednoho okna. Ukázka způsobu zadání parametrů je na obr. 2.9. Při použití regulátoru typu P, musí být zadána alespoň velikost zesílení K, ale ostatní konstanty nejsou potřeba. Velikost zesílení K je poté vždy první hodnota v řetězci. Pro PS regulátor je potřeba zadat nejméně dvě hodnoty, kde první

hodnota je zesílení K a druhá hodnota je velikost integrační konstanty T_I. Při použití regulátoru typu PD se musí zadat nejméně tři hodnoty. Pokud zadáme jen tři hodnoty, tak první hodnota je velikost zesílení K, druhá hodnota je velikost derivační konstanty T_D a třetí hodnota je velikost omezení zesílení pro vyšší frekvence N. Při zadání všech čtyř parametrů je první hodnota velikost zesílení K, druhá hodnota je velikost integrační konstanty T_I, která ale nebude použita, třetí hodnota je velikost derivační konstanty T_D a čtvrtá hodnota je velikost omezení zesílení pro vyšší frekvence N. Předchozí věta platí i pro použití regulátoru typu PSD, kde se musí zadat všechny čtyři hodnoty.



Obrázek 2.8: Výběr typu regulátoru



Obrázek 2.9: Nastavení parametrů prvního regulátoru

2.3.3 Vzorkovací perioda

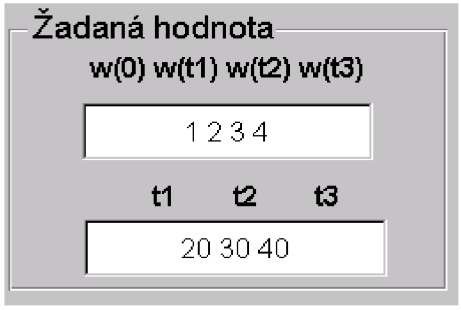
Vzorkovací perioda je ve výchozím stavu nastavena na velikost jedné pětiny reálné části největší časové konstanty výchozího stavu soustavy. Při prvním spuštění je poté zobrazena v části vzorkování. Příklad nastavené vzorkovací periody je na obr. 2.10. V tomto případě je nastavena na velikost $T_s = 0,2$ s. Uživatel si může velikost vzorkovací periody nastavit podle svých potřeb tím, že změní hodnotu z 0 na hodnotu, kterou požaduje. Zadávaná

hodnota je v sekundách. Pokud chce poté použít výchozí nastavení, stačí změnit hodnotu zpátky na hodnotu 0.

Obrázek 2.10: Velikost vzorkovací periody

2.3.4 Velikost žádané hodnoty

Program umožňuje třikrát změnit velikost žádané hodnoty. Žádaná hodnota se mění v předem definovaném čase, který zadal uživatel tohoto programu. Pokud je definován čas změny a tento čas je menší než délka simulace, musí být definována velikost žádané hodnoty, jinak program zahlásí chybu. Příklad zadání více žádaných hodnot je na obr. 2.11.



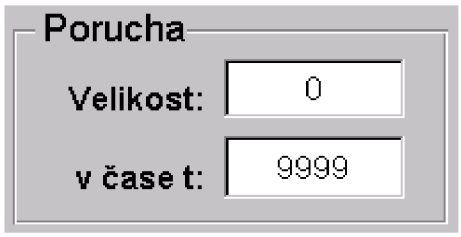
Obrázek 2.11: Nastavení žádané hodnoty

2.3.5 Nastavení poruchy

Program umožňuje otestovat použitý PSC systém s poruchovým signálem, který vstupuje do soustavy a je přičítán k akčnímu zásahu z regulátoru. U poruchového signálu se nastaví velikost a čas, kdy porucha začne působit. Okno pro nastavení poruchy je na obr. 2.12.

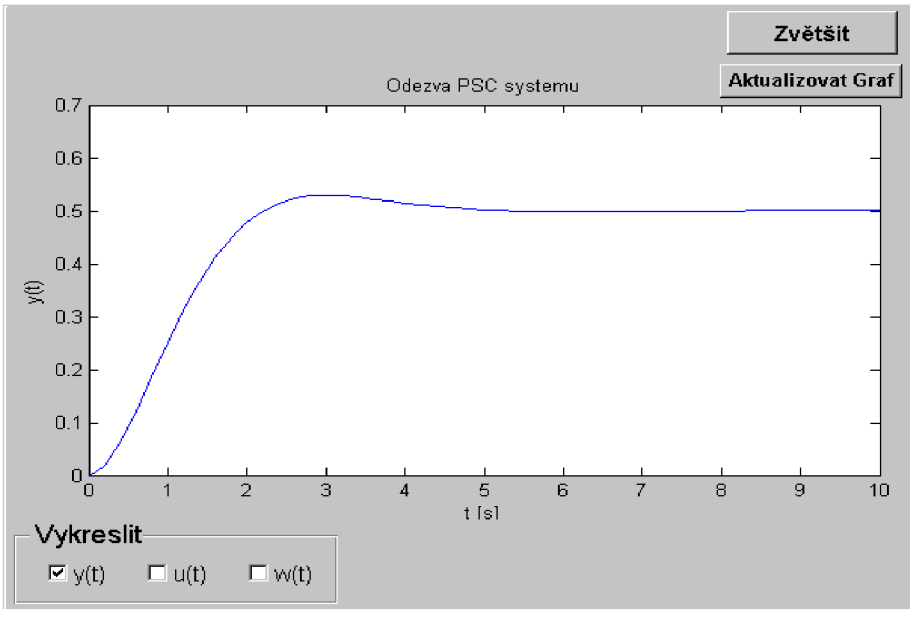
2.3.6 Zobrazení výsledků simulace

Výsledný přechodový děj se zobrazí po simulaci v grafu zobrazeném v grafickém uživatelském prostředí. Příklad vykreslení přechodového děje je na obr. 2.13. Program umožňuje



Obrázek 2.12: Nastavení poruchy

vykreslit průběh výstupní veličiny $y(t)$, žádané veličiny $w(t)$ a akční veličiny regulátoru $u(t)$. Výstupní veličina $y(t)$ je zobrazena modrou barvou, žádaná veličina $w(t)$ je zobrazena červenou barvou a akční veličina regulátoru $u(t)$ je zobrazena zelenou barvou. Program ukládá velikost těchto veličin v každém kroku do matice, takže je možné si zobrazit například akční zásah z regulátoru aniž by se musela znovu opakovat simulace. Zobrazení další veličiny se docílí zaškrtnutím požadované veličiny a stisknutím tlačítka *Aktualizovat Graf*. Poté se do grafu vykreslí všechny požadované veličiny. Pokud potřebuje uživatel zvětšit zobrazený průběh a dále si ho upravit, může použít tlačítko *Zvětšit*, které zobrazí nové okno s průběhem nebo průběhy, kde jsou tyto možnosti zpřístupněny.



Obrázek 2.13: Zobrazení výsledků simulace

2.3.7 Chybějící obrázek loga FETK

Jedním z devíti souborů je obrázek loga Fakulty elektrotechniky a komunikačních technologií. Tento obrázek musí být společně s ostatními soubory v místě, kde má program MATLAB nastavenou aktuální pracovní složku (Current directory). Při jeho absenci dojde při spuštění programu k chybě, která zamezí správné funkčnosti programu a program se stává nefunkční. Nicméně tento soubor není pro simulaci nijak důležitý, a tak jednoduchou úpravou souboru *pscsys.m* lze tuto chybu odstranit. Opravení chyby při spuštění je zobrazeno na příkladu 2.7 a 2.8, kde je zobrazena část kódu souboru *pscsys.m* na řádku 59 před úpravou (příklad 2.7) a po úpravě (příklad 2.8).

Příklad 2.7: Načtení loga FEKT

```
imshow('logo_fekt.png')
```

Příklad 2.8: Zamezení načtení loga FEKT

```
%imshow('logo_fekt.png')
```

Úprava spočívá v přidání příkazu v příkladu 2.7 do komentáře a tím zamezení načtení chybějícího souboru.

2.4 Zpracování dat z GUI

Při vytvoření grafického uživatelského rozhraní pomocí nástroje GUIDE se vytváří soubor typu m-file, který obsahuje funkce ovládající grafické uživatelské rozhraní. Částečný popis funkcí základních prvků je v části 2.2.

2.4.1 Globální proměnné

Soubor *pscsys.m* obsahuje pět globálních proměnných, které jsou společné pro všechny funkce. Tyto proměnné jsou zobrazeny na příkladě 2.9. Všechny tyto globální proměnné jsou součástí objektu *handles* a pro přístup k nim se používá tečka, jak je ukázáno na příkladě 2.9.

Příklad 2.9: Globální proměnné v souboru pscsys.m

```
handles.type = 'P';
handles.prepinac=1;
handles.ScopeDat=0;
handles.CheckBox=[1 0 0];
```

Proměnná *handles.type* slouží pro předání typu regulátoru do dalšího zpracování. Proměnná *handles.prepinac* slouží pro detekci chyby při zpracování programu. Tato chyba nastane důsledkem špatně zadaných dat a tato proměnná se nastaví na hodnotu 99. Nastavení proměnné *handles.prepinac* na hodnotu 99 má význam pro další zpracování programu, protože některé části programu jsou zpracovávány jen v případě, že nenastala chyba. Díky tomu je možné zobrazit chybové hlášení s chybou a jejím popisem, aniž by nastala chyba při zpracovávání programu programem MATLAB, která by měla za následek nemožnost zobrazení chybové zprávy. Proměnná *handles.ScopeDat* slouží pro uložení matice výsledných veličin, kvůli možnosti zvětšení a aktualizace grafu. Poslední proměnná *handles.CheckBox* slouží jako pomocná proměnná obsahující informaci o požadovaných veličinách, které chce uživatel tohoto programu vykreslit.

2.4.2 Tlačítko START

Tlačítko START, které je na obr. 2.3, slouží pro spuštění simulace. Při jeho stisku se začnou zpracovávat požadovaná data a následně proběhne simulace. Zpracování dat probíhá převodem řetězců na čísla a uložení do lokálních proměnných, které je možné použít v dalším zpracování. Toto je zobrazeno na příkladu 2.10, kde *numerator1* je název prvku Text Edit do kterého se zadává čitatel počátečního stavu soustavy. Následně je zavolán soubor *PSC_system_config.m*. Poté je zobrazena použitá velikost vzorkovací periody příkazem v příkladě 2.11, kde *pomst* je proměnná obsahující velikost vzorkovací periody ve formátu řetězce a *PSC_system_config.m* je název prvku textového pole, které bude změněno na tuto hodnotu. Toto pole je vidět na obr. 2.10. Nakonec je zde kód zobrazující výsledný graf v příslušné části grafického uživatelského prostředí.

Příklad 2.10: Funkce převádějící čitatel počátečního stavu soustavy na pole čísel

```
num1 = str2num(get(handles.numerator1,'String'));
```

Příklad 2.11: Funkce zobrazující velikost použité vzorkovací periody

```
set(handles.SamplingTimeTxt,'String',pomst);
```

2.5 Příprava dat pro simulaci

Po zpracování dat z grafického uživatelského prostředí se spustí soubor *PSC_system_config.m*, který má za úkol získaná data uspořádat do jednotlivých matic, které se pak předají simulaci.

2.5.1 Data pro soustavu - sfunsys.m

V první části se zjišťuje počet stavů, kterých soustava nabývá a jestli změna stavů nastává ve správném pořadí. Poté se vytvoří vektor *T*, který obsahuje informace o časech, kdy má nastat změna stavu soustavy. Další část testuje, jestli byly splněny limitující podmínky simulace. Řád soustavy není větší než pět a menší než dva a řád čitatele je menší než řád jmenovatele. Poté je kontrola, jestli není nějaký čitatel roven nule a jestli se nemění řád soustavy při přechodu z jednoho stavu do druhého. Pak se vypočítává velikost vzorkovací periody, pokud nebyla uživatelem zadána. Velikost vzorkovací periody je uložena do proměnné *Tss*. Po vypočtení vzorkovací periody se přepočítají prvky matice *T* tak, aby změna stavu soustavy nastala v n-násobek vzorkovací periody. Následuje diskretizace jednotlivých stavů a uložení koeficientů z polynomu čitatele a jmenovatele do matice *Spom*. Tato matice je pak upravena na vhodný tvar do matice *Sroot*.

Příklad matice *SRoot* pro soustavu druhého řádu a vektor *T*. Koeficienty jsou získány z tvaru rovnice 1.10.

$$\mathbf{SRoot} = \begin{pmatrix} b1_2 & b1_1 & b1_0 & 0 & 0 & 0 \\ a1_2 & a1_1 & a1_0 & 0 & 0 & 0 \\ b2_2 & b2_1 & b2_0 & 0 & 0 & 0 \\ a2_2 & a2_1 & a2_0 & 0 & 0 & 0 \\ b3_2 & b3_1 & b3_0 & 0 & 0 & 0 \\ a3_2 & a3_1 & a3_0 & 0 & 0 & 0 \\ b4_2 & b4_1 & b4_0 & 0 & 0 & 0 \\ a4_2 & a4_1 & a4_0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{T} = \begin{pmatrix} 0 & t_1 & t_2 & t_3 \end{pmatrix}$$

2.5.2 Data pro regulátor - sfunPSD.m

Zpracování dat pro regulátor začne jen v případě, že nenastala žádná chyba při zpracování dat pro soustavu. Zpracování dat probíhá pomocí přepínače, který vybere typ regulátoru a pak probíhá kontrola, jestli je zadán dostatečný počet konstant. Všechny možnosti způsobu zadání konstant pro jednotlivé typy regulátorů jsou uvedeny v části 2.3.2 Nastavení regulátoru. Poté je vytvořena matice *SReg*. V programu je provedeno beznárazové přepínání mezi jednotlivými tvary adaptivního regulátoru pro regulátory typu PS a PSD. Při změně tvaru regulátoru je velikost sumační složky regulátoru nastavena na velikost odpovídající převrácené hodnotě statického zesílení stavu soustavy, pro kterou je navržen příslušný tvar regulátoru. Velikosti, které se nastaví pro každou sumační složku jsou uloženy ve vektoru *Iinit*.

$$\mathbf{SReg} = \begin{pmatrix} K_1 & K_2 & K_3 & K_4 \\ T_{I1} & T_{I2} & T_{I3} & T_{I4} \\ T_{D1} & T_{D2} & T_{D3} & T_{D4} \\ N_1 & N_2 & N_3 & N_4 \end{pmatrix}$$

$$\mathbf{Iinit} = \begin{pmatrix} I_{init1} & I_{init2} & I_{init3} & I_{init4} \end{pmatrix}$$

2.5.3 Data pro změnu žádané hodnoty - sfunw.m

Zpracování dat pro změnu žádané hodnoty začne jen v případě, že nenastala žádná chyba v předešlém zpracování dat. Při zpracování dat pro žádanou hodnotu se kontroluje, jestli je pro každý čas změny žádané hodnoty menší než čas simulace přiřazena hodnota žádaná. Získaná data jsou uložena do matice *Wmat*.

$$\mathbf{Wmat} = \begin{pmatrix} w(0) & w(t_{w1}) & w(t_{w2}) & w(t_{w3}) \\ 0 & t_{w1} & t_{w2} & t_{w3} \end{pmatrix}$$

2.5.4 Ostatní data

- Porucha - Pro simulaci PSC systému je zvolen pevný krok simulace o velikosti vzorkovací periody. Toto je možné z důvodu diskrétní soustavy a regulátoru a výsledkem

je zrychlení simulace, protože program MATLAB nemusí počítat stavy mezi jednotlivými vzorky. Proto čas poruchy je přepočítán tak, aby nastal v co nejbližším kroku.

- Identifikace - V této části je nastavena proměnná *HowOften*, která značí, jak často se bude vyhodnocovat změna stavu soustavy.

- Simulace - Nakonec je zde nastavení vlastností simulace a spuštění simulace. Po simulaci se do matice *ScopeDPSC* přidají velikosti hodnot akčního zásahu a žádané hodnoty v každém časovém okamžiku.

$$\textbf{ScopeDPSC} = \begin{pmatrix} t(0) & y(0) & u(0) & w(0) \\ t(T_s) & y(T_s) & u(T_s) & w(T_s) \\ t(2T_s) & y(2T_s) & u(2T_s) & w(2T_s) \\ \vdots & \vdots & \vdots & \vdots \\ t(nT_s) & y(nT_s) & u(nT_s) & w(nT_s) \end{pmatrix}$$

2.6 Schéma modelu

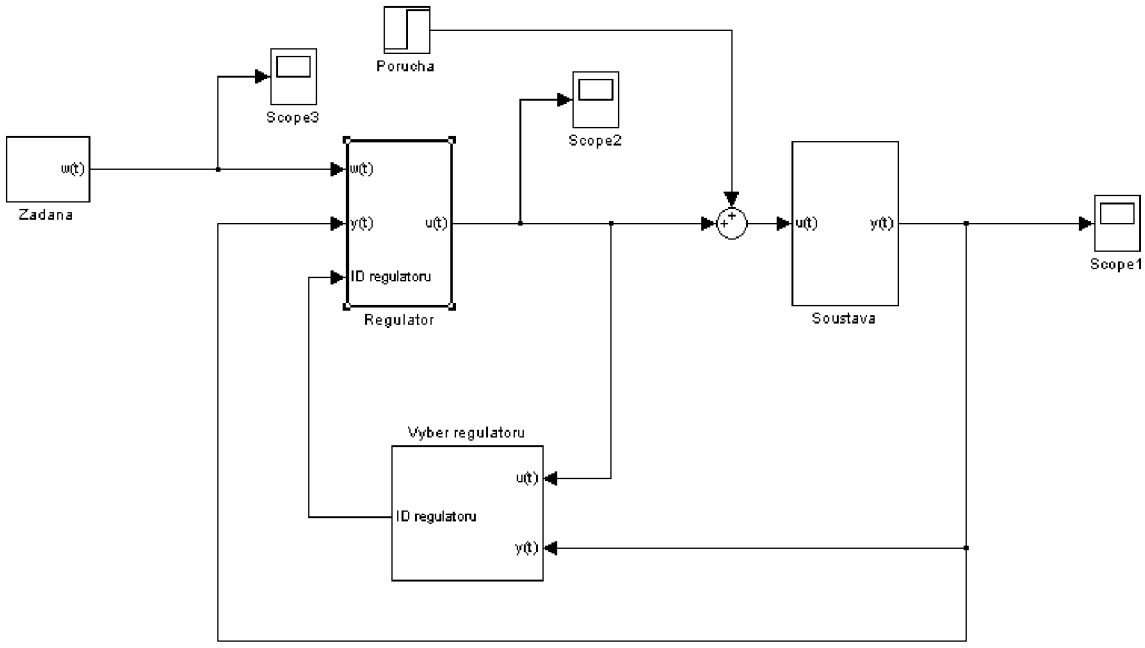
Model systému typu PSC je vytvořen v nadstavbě Simulink programu MATLAB. Model se skládá ze soustavy, adaptivního regulátoru a bloku pro identifikaci, z kterého vychází řídicí proměnná značící číslo stavu soustavy, které se má regulátor přizpůsobit. Schéma modelu systému typu PSC je na obr. 2.14.

Při tvorbě modelu vznikl problém s algebraickou smyčkou, která obsahovala regulátor, soustavu a zpětnou vazbu. Tento problém se povedl vyřešit přidáním zpoždění o velikosti jedné vzorkovací periody k regulátoru. Aby byl ale dodržen přenos přímé větve, musí se čitatel soustavy posunout o jednu vzorkovací periodu. Této úpravy je možné docílit proto, protože aktuální výstup diskrétní soustavy není závislý na velikosti aktuálního vstupu.

$$F_0(z^{-1}) = F_R(z^{-1})F_S(z^{-1}) = F_R(z^{-1})z^{-1}F_{S2}(z^{-1})$$

$$F_0(z^{-1}) = \frac{d_0+d_1z^{-1}+d_2z^{-2}}{1+c_1z^{-1}+c_2z^{-2}}\frac{b_1z^{-1}+b_0z^{-2}}{a_2z^0+a_1z^{-1}+a_0z^{-2}} = \frac{d_0+d_1z^{-1}+d_2z^{-2}}{1+c_1z^{-1}+c_2z^{-2}}z^{-1}\frac{b_1z^0+b_0z^{-1}}{a_2z^0+a_1z^{-1}+a_0z^{-2}}$$

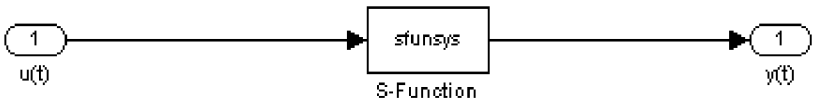
kde F_0 je přenos přímé větve, F_R je přenos regulátoru, F_S je přenos soustavy a F_{S2} je přenos soustavy s posunutým čitatelem o jednu vzorkovací periodu dopředu.



Obrázek 2.14: Schéma modelu systému typu PSC

2.6.1 Blok Soustava

Blok Soustava obsahuje jednu s-funkci sfunsys, která zastupuje soustavu, jak je zobrazeno na obr. 2.15. Vstupem do bloku Soustava je akční zásah regulátoru a výstupem z tohoto bloku je výstupní hodnota soustavy.



Obrázek 2.15: Blok Soustava

Hlavička této s-funkce je zobrazena na příkladu 2.12. Této s-funkci jsou předávány tři parametry. Prvním parametrem je matice *Sroot*. Druhým parametrem je vektor *T*. Více o těchto maticích v části 2.5.1 Data pro soustavu - sfunsys.m. Posledním parametrem je velikost vzorkovací periody. Tato s-funkce obsahuje jeden vstup a jeden výstup.

Příklad 2.12: Hlavička s-funkce sfunsys

```
function [sys,x0, str, ts] = sfunsys(t,x,u, flag, Sroot,T,Tss)
```

V části s-funkce sfunsys *mdlOutputs* probíhá výpočet aktuální hodnoty výstupu. Nejdříve se zjišťuje, jestli nemá nastat změna stavu soustavy. Pokud ano, zvětší se pomocná proměnná *ind*, která je indexem řádku v matici koeficientů a při výpočtu aktuální hodnoty výstupu se použijí koeficienty požadovaného stavu. Matematický zápis zdrojového kódu výpočtu aktuální hodnoty výstupu pro první stav soustavy druhého řádu z příkladu 2.13 je

$$\mathbf{y(k)} = \begin{pmatrix} b1_1 & b1_0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u(k) \\ u(k-1) \\ u(k-2) \\ u(k-3) \\ u(k-4) \end{pmatrix} - \begin{pmatrix} a1_1 & a1_0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} y(k-1) \\ y(k-2) \\ y(k-3) \\ y(k-4) \\ y(k-5) \end{pmatrix}.$$

Příklad 2.13: Výpočet aktuální hodnoty výstupu

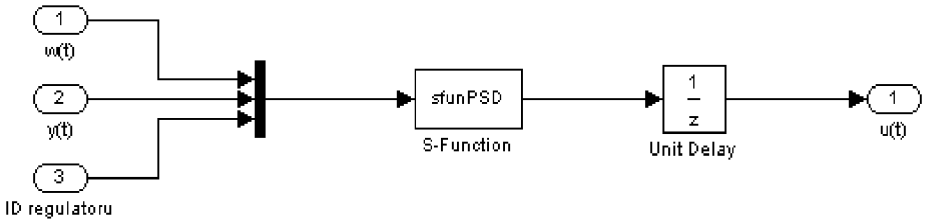
```
ys(1)=Sroot(ind,2:6)*us'-Sroot(ind+1,2:6)*ys(2:6)';
```

Z matematického zápisu lze vidět, že výpočet aktuální hodnoty výstupu je závislý na aktuální hodnotě vstupu. To je z důvodu přidání dopravního zpoždění o velikosti jedné vzorkovací periody do přímé větve. Nakonec se aktualizují vektory obsahující informaci o hodnotách vstupu a výstupu pro výpočet v dalším kroku.

2.6.2 Blok Regulátor

Blok Regulátor obsahuje jednu s-funkci sfunPSD, která zastupuje adaptivní regulátor a dopravní zpoždění o velikosti jedné vzorkovací periody. Toto je zobrazeno na obr. 2.16. Hlavička této s-funkce je zobrazena na příkladu 2.14. Vstupem do bloku Regulátor je žádaná hodnota, výstupní hodnota soustavy a číslo regulátoru, který má být použit. Výstupem z tohoto bloku je akční zásah regulátoru.

Hlavička této s-funkce je zobrazena na příkladu 2.14. Této s-funkci jsou předávány čtyři parametry. Prvním parametrem je velikost vzorkovací periody. Druhým parametrem je matice *SReg*. Třetím parametrem je požadovaný typ regulátoru a posledním parametrem je vektor *Iinit*. Tato s-funkce obsahuje tři vstupy a jeden výstup.



Obrázek 2.16: Blok Regulátor

Příklad 2.14: Hlavička s-funkce sfunPSD

```
function [sys,x0,str,ts] = sfunPSD(t,x,u,flag,Ts,SReg,reg_type,Iinit)
```

V části s-funkce sfunPSD *mdlOutputs* probíhá výpočet velikosti akčního zásahu. Nejdříve se nastaví parametry regulátoru, které byly vybrány pro zvolený stav soustavy. Poté se vypočte velikost regulační odchylky. V dalším kroku se kontroluje, jestli nebyly změněny parametry regulátoru kvůli beznárazovému přepínání. Pokud parametry regulátoru byly změněny, nastaví se velikost sumační složky na hodnotu, která byla vypočtena pro příslušný stav soustavy. Nakonec následuje výběr požadovaného typu regulátoru a výpočet velikosti akčního zásahu. Tento výpočet je odvozen v části 1.2.4 PSD regulátor.

2.6.3 Blok Výběr regulátoru

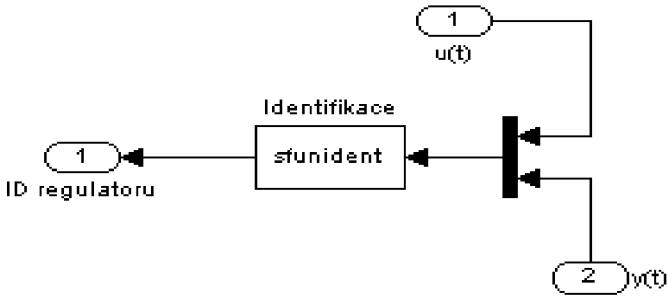
Blok Výběr regulátoru obsahuje jednu s-funkci sfunident, jak je zobrazeno na obr. 2.17. Vstupem do bloku Výběr regulátoru je akční zásah regulátoru a výstupní hodnota soustavy. Výstupem z tohoto bloku je řídicí proměnná, která je v tomto případě číslo regulátoru.

Hlavička této s-funkce je zobrazena na příkladu 2.15. Této s-funkci jsou předávány tři parametry. Prvním parametrem je matice *SRoot*. Druhým parametrem je velikost vzorkovací periody. Třetím parametrem je informace, jak často se má provádět identifikace soustavy. Tato s-funkce obsahuje dva vstupy a jeden výstup.

Příklad 2.15: Hlavička s-funkce sfunident

```
function [sys,x0,str,ts] = sfunident(t,x,u,flag,Sroot,Tss,HowOften)
```

V části s-funkce sfunident *mdlOutputs* probíhá výběr regulátoru. Nejdříve se vypočtou výstupní hodnoty všech modelů obsažených v matici *SRoot*. Následuje výpočet odchylky

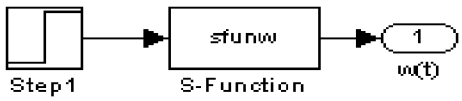


Obrázek 2.17: Blok Výběr regulátoru

každé výstupní hodnoty modelu od výstupní hodnoty soustavy. Z těchto odchylek se vybere nejmenší a ta zároveň určí číslo regulátoru, které se nastaví na výstup této s-funkce.

2.6.4 Blok Žádaná

Blok Žádaná obsahuje jednu s-funkci sfunw a jednotkový skok v čase 0 s, jak je zobrazeno na obr. 2.18. Blok Žádaná neobsahuje žádnou vstupní veličinu. Výstupem z tohoto bloku je žádaná hodnota.



Obrázek 2.18: Blok Žádaná

Hlavička této s-funkce je zobrazena na příkladu 2.16. Této s-funkci jsou předávány dva parametry. Prvním parametrem je matice *SWmat*. Druhým parametrem je velikost vzorkovací periody. Tato s-funkce neobsahuje žádný vstup a obsahuje jeden výstup.

Příklad 2.16: Hlavička s-funkce sfunw

```
function [sys,x0, str , ts ] = sfunw(t,x,u, flag ,Wmat,Tss)
```

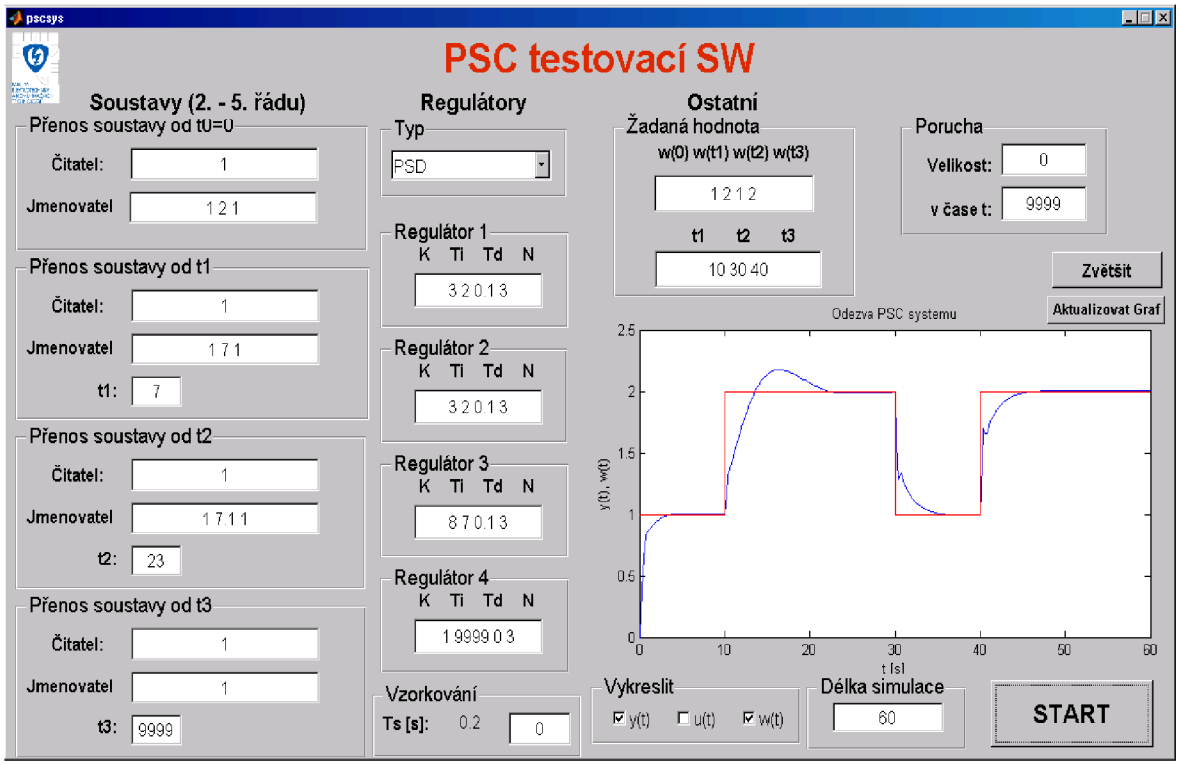
V části s-funkce sfunw *mdlOutputs* probíhá pouze změna velikosti žádané hodnoty. Nastane-li požadavek na změnu žádané hodnoty, změní se zesílení signálu jednotkového skoku na požadovanou velikost.

3 DEMONSTRACE PROGRAMU

V následující části bude ukázka vybraných funkcí programu.

3.1 Demonstrace 1

Cílem této demonstrace je ukázat změnu žádané hodnoty a použití jiných parametrů regulátoru pro podobný stav soustavy a jejich vliv na přechodný děj. Nastavení simulace je vidět na obr. 3.1.



Obrázek 3.1: Demonstrace 1: Nastavení

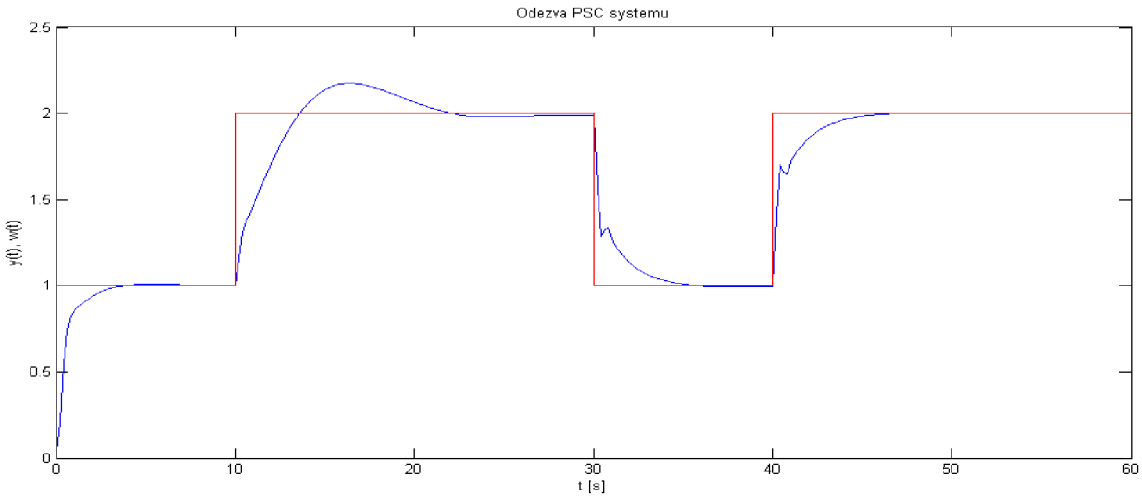
V demonstraci jsou použity tři stavy soustavy. Regulátor je typu PSD. Vzorkovací perioda není nastavena a vypočtená vzorkovací perioda je $T_s = 0,2$ s. Žádaná hodnota je $w(0) = 1$, od $t_1 = 10$ s je $w(t_1) = 2$, od $t_2 = 30$ s je $w(t_2) = 1$ a od $t_3 = 40$ s je $w(t_3) = 2$.

$$F_{S1}(p) = \frac{1}{p^2 + 2p + 1} \qquad F_{R1} = 3\left(1 + \frac{1}{2p} + \frac{0.1p}{\frac{0.1}{3}p + 1}\right)$$

$$F_{S2}(p) = \frac{1}{p^2+7p+1} \qquad F_{R2} = 3\left(1+\frac{1}{2p}+\frac{0.1p}{\frac{0.1}{3}p+1}\right)$$

$$F_{S3}(p) = \frac{1}{p^2+7.1p+1} \qquad F_{R3} = 8\left(1+\frac{1}{7p}+\frac{0.1p}{\frac{0.1}{3}p+1}\right)$$

Zvětšený přechodový děj je vidět na obr. 3.2.



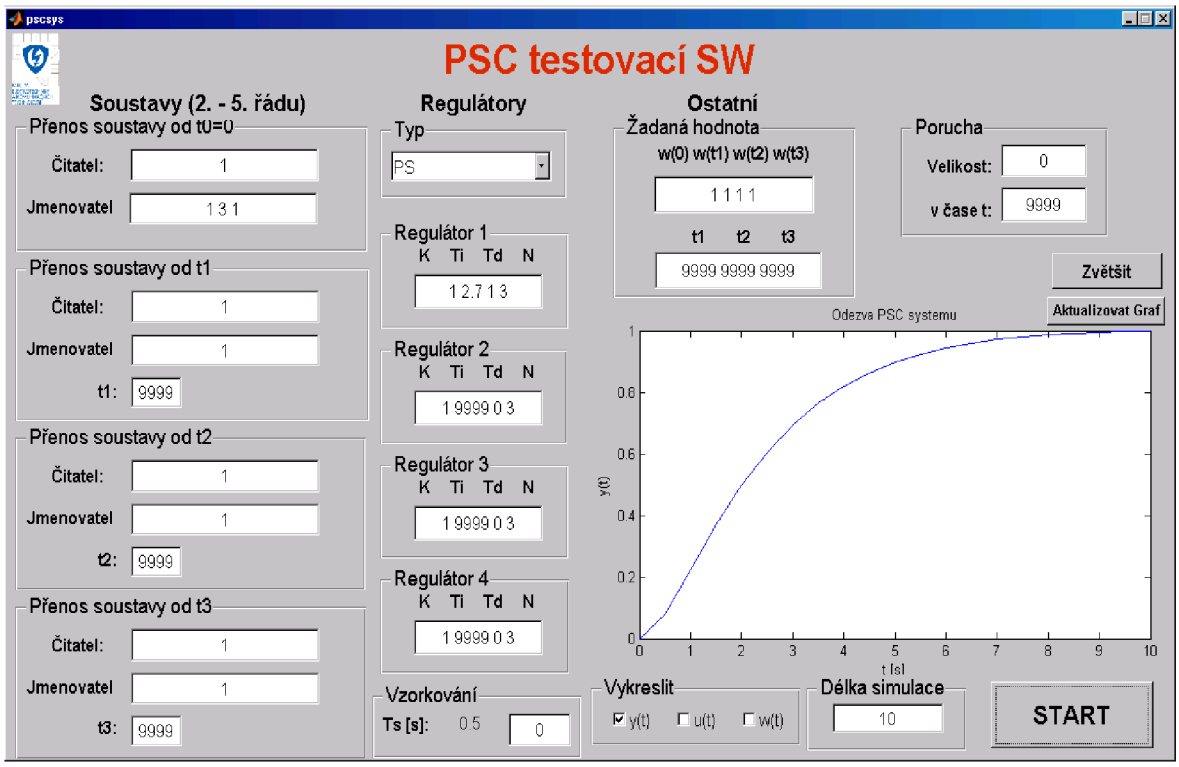
Obrázek 3.2: Demonstrace 1: Graf

3.2 Demonstrace 2

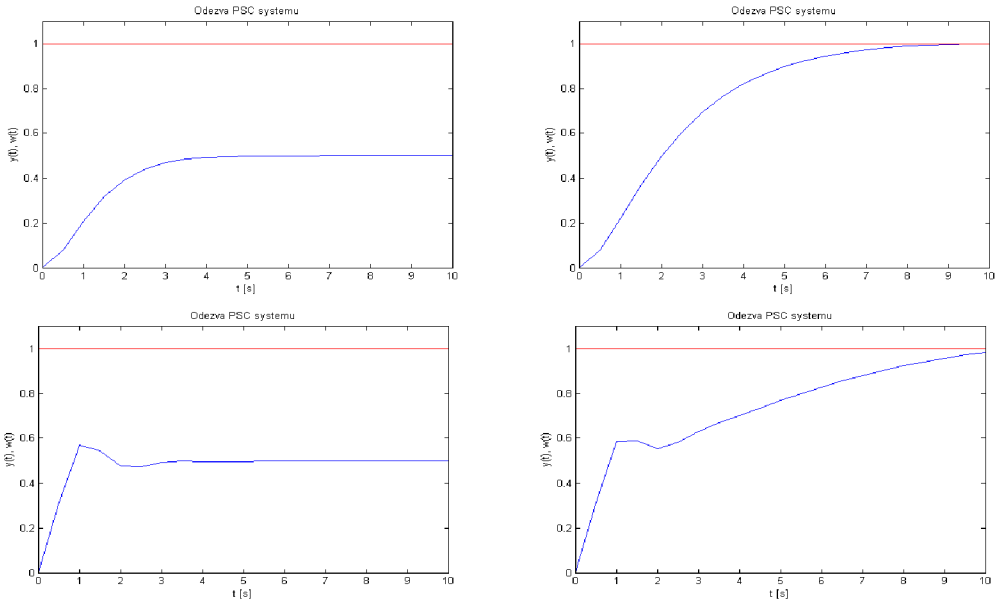
Cílem této demonstrace je ukázat vliv typu regulátoru se stejnými parametry na přechodný děj. Nastavení simulace je vidět na obr. 3.3.

V demonstraci je použit jeden stav soustavy. Regulátor je typu P, PS, PD a PSD. Vzorkovací perioda není nastavena a vypočtená vzorkovací perioda je $T_s = 0,5$ s. Žádaná hodnota je $w = 1$. Porovnání přechodového děje pro různé přenosy regulátoru F_R je na obr. 3.4.

$$F_S(p) = \frac{1}{p^2+3p+1} \qquad F_R = 3\left(1+\frac{1}{2.7p}+\frac{p}{\frac{1}{3}p+1}\right)$$



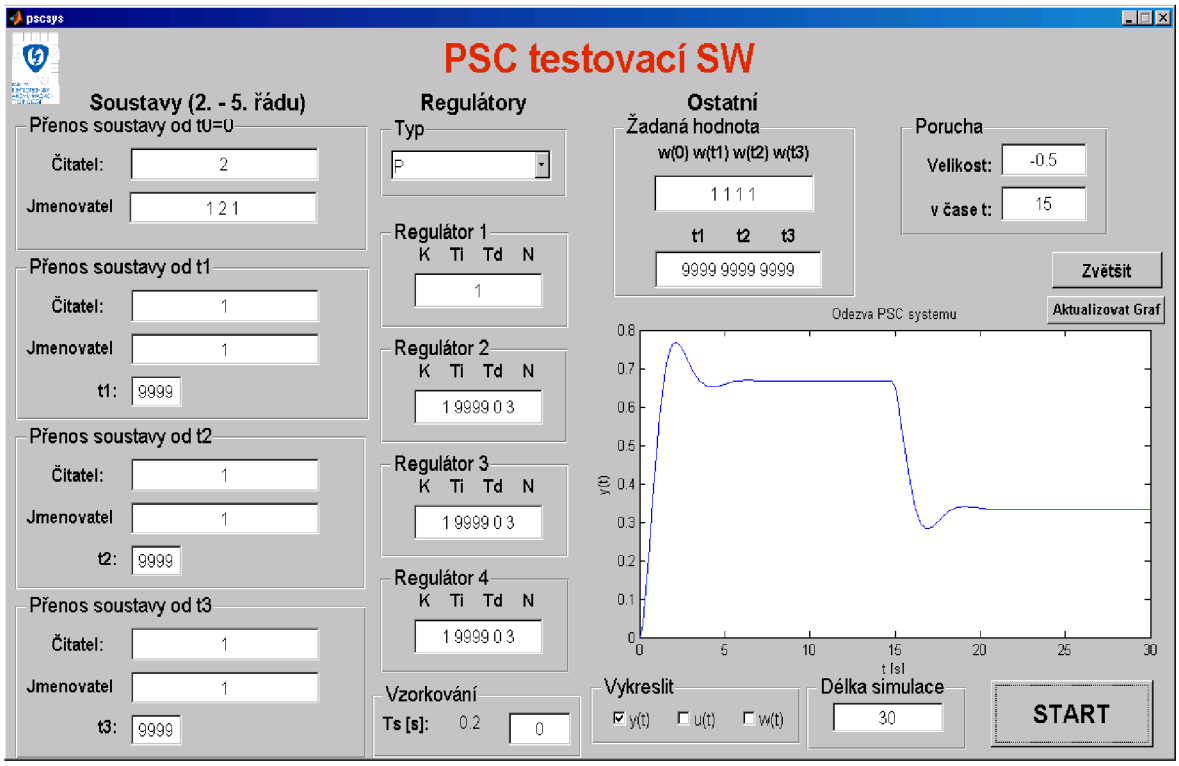
Obrázek 3.3: Demonstrace 2: Nastavení



Obrázek 3.4: Porovnání přechodového děje pro P, PS, PD a PSD regulátor, kde vlevo nahoře je P regulátor, vpravo nahoře je PS regulátor, vlevo dole je PD regulátor a vpravo dole je PSD regulátor

3.3 Demonstrace 3

Cílem této demonstrace je ukázat vliv poruchy na přechodný děj. Nastavení simulace je vidět na obr. 3.5.

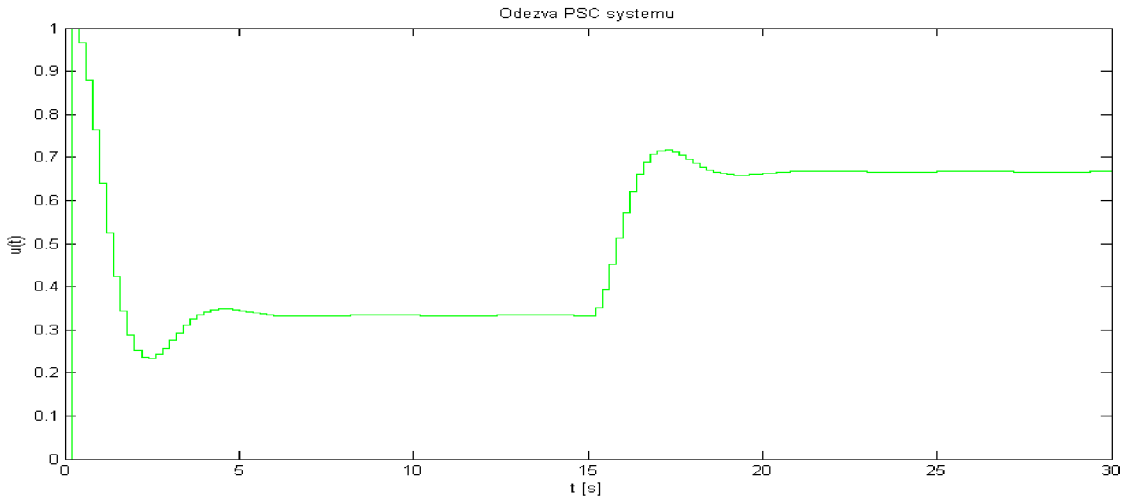


Obrázek 3.5: Demonstrace 3: Nastavení

V demonstraci je použit jeden stav soustavy. Regulátor je typu P. Vzorkovací perioda není nastavena a vypočtená vzorkovací perioda je $T_s = 0,2$ s. Žádaná hodnota je $w(0) = 1$. Poruchový signál začne působit v čase $t_P = 15$ s a jeho velikost je -0.5.

$$F_{S1}(p) = \frac{2}{p^2 + 2p + 1} \qquad F_R = 1$$

Průběh akční veličiny $u(t)$ je zvětšen na obr. 3.6.

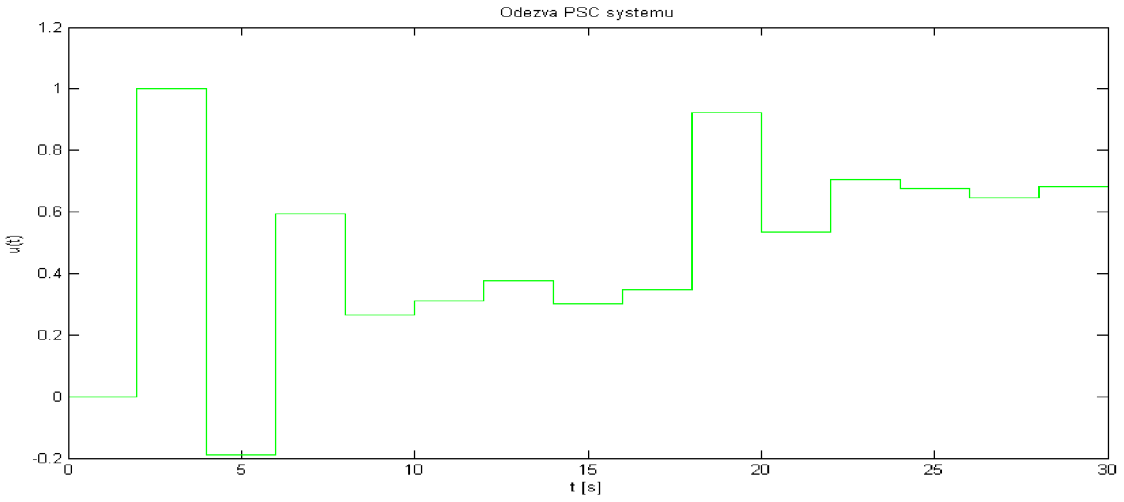


Obrázek 3.6: Demonstrace 3: Graf

3.4 Demonstrace 4

Cílem této demonstrace je ukázat vliv velikosti vzorkovací periody na přechodný děj. Nastavení této demonstrace je stejné jako v předchozí demonstraci, ale je zde nastavena vzorkovací perioda na velikost $T_s = 1$ s.

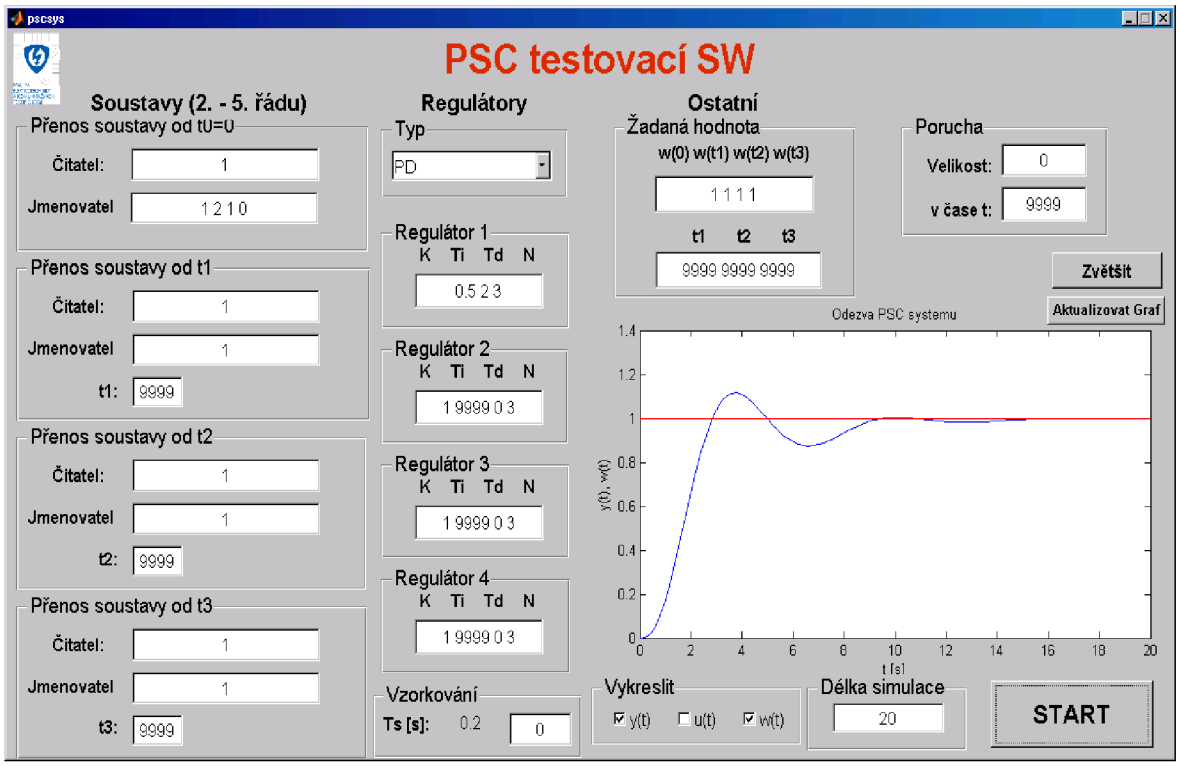
Průběh akční veličiny pro porovnání s průběhem akční veličiny na obr. 3.6 je zvětšen na obr. 3.7.



Obrázek 3.7: Demonstrace 4: Graf

3.5 Demonstrace 5

Cílem této demonstrace je ukázat možnost simulace soustavy s astatismem. Nastavení simulace je vidět na obr. 3.8.

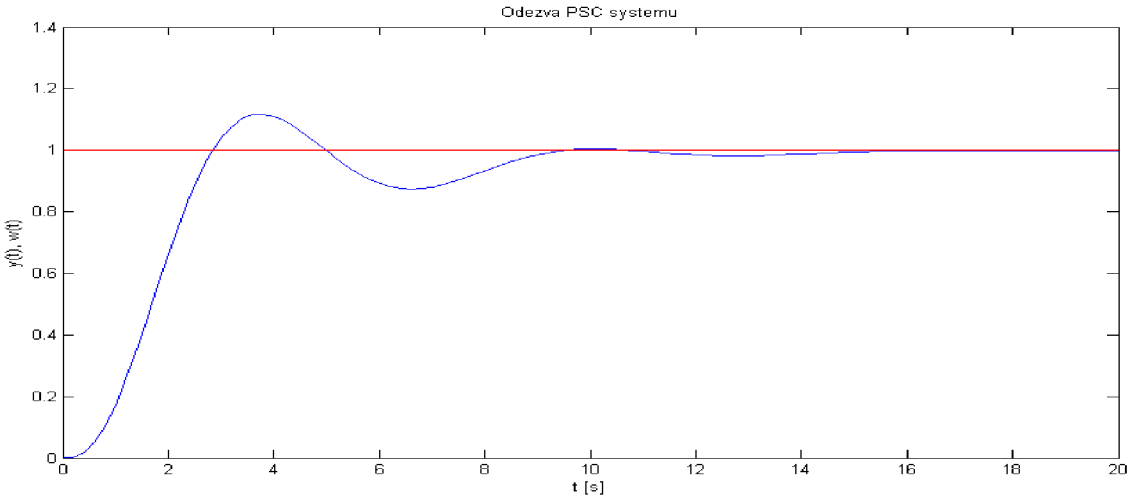


Obrázek 3.8: Demonstrace 5: Nastavení

V demonstraci je použit jeden stav soustavy. Regulátor je typu PD. Vzorkovací perioda není nastavena a vypočtená vzorkovací perioda je $T_s = 0,2$ s. Žádaná hodnota je $w(0) = 1$.

$$F_S(p) = \frac{1}{p(p^2+2p+1)} \qquad F_R = 0.5\left(1+\frac{2p}{\frac{2}{3}p+1}\right)$$

Zvětšený přechodový děj je vidět na obr. 3.9.



Obrázek 3.9: Demonstrace 5: Graf

4 ZÁVĚR

Cílem této práce bylo vytvořit program na testování adaptivních systémů typu PSC. Tento program se ovládá pomocí grafického uživatelského prostředí, které je intuitivní a uživatel se v něm rychle zorientuje.

Změna parametrů soustavy je provedena pomocí skokové změny přenosové funkce soustavy v nastaveném čase. Nevýhodou této změny je její fyzikální nerealizovatelnost. Tento problém by se dal vyřešit postupnou změnou parametrů.

Stavy soustavy se zadávají ve spojitém tvaru, ale při simulaci se počítá s diskrétními tvary. Výhodou použití diskrétních tvarů je možnost simulace soustav s komplexními kořeny. Nevýhodou je, že se musí použít vhodná vzorkovací perioda, která ale nemusí být vždy vhodně nastavena.

Jako vylepšení této práce by bylo použití nelineární soustavy místo změny stavů soustavy použitých v této práci. Při použití nelineární soustavy by bylo ale vhodné použít jinou řídicí proměnnou, než je výstup z identifikace použitý v této práci. Další možností rozšíření této práce by bylo rozšíření počtu typů regulátorů a možnost nastavení omezení akčního zásahu.

Správnost simulovaného systému byla ověřena v programu MATLAB na jednom stavu soustavy s regulátorem, které byly zadány ve spojitém tvaru. Poté diskretizovány funkcí *c2d*. Následně byl vypočten přenos řízení $F_W = \frac{F_R F_S}{1+F_R F_S}$ a vypočtena odezva na jednotkový skok pomocí příkazu *step*(F_W). Vypočtená a simulovaná odezva byla identická.

REFERENCE

[1] ÅSTRÖM, K. J. - WITTENMARK, B. *Adaptive Control*. 2. vyd. London: Prentice Hall, 1994. 574 s.

[2] SASTRY, S. - BODSON, M. *Adaptive Control: Stability, Convergence, and Robustness*. 1. vyd. New Jersey: Prentice Hall, 1989. 377 s. ISBN 0-13-004326-5

[3] PIVOŇKA, P. *Číslicová řídicí technika*. Skriptum VUT. Brno: VUT-FEKT, Ústav automatizace a měřicí techniky, 2003, 151 s.

[4] DUŠEK, F. *MATLAB a SIMULINK*. Skriptum UPCE. Pardubice. 2000, 146 s.

[5] ŠOLC, F. - VÁCLAVEK, P. *Modelování a simulace*. Verze 0.11 Skriptum VUT. Brno: VUT-FEKT, Ústav automatizace a měřicí techniky, 2007, 169 s.

[6] BLAHA, P. - VAVŘÍN, P. *Řízení a regulace I: Základy regulace lineárních systémů - spojité a diskrétní*. Skriptum VUT. Brno: VUT-FEKT, Ústav automatizace a měřicí techniky, 215 s.

[7] ŠOLC, F. - VÁCLAVEK, P. - VAVŘÍN, P. *Řízení a regulace II: Analýza a řízení nelineárních systémů*. Verze 1.7. Skriptum VUT. Brno: VUT-FEKT, Ústav automatizace a měřicí techniky, 2007, 228 s.

[8] JURA, P. - POLANSKÝ, M. *Signály a systémy: Část 3: Diskrétní signály a diskrétní systémy*. Skriptum VUT. Brno: VUT-FEKT, Ústav automatizace a měřicí techniky, 87 s.

[9] QUACH, Q. *MATLAB GUI (Graphical User Interface) Tutorial for Beginners* [online]. 23. 10. 2007 [cit. 5. 5. 2010]. Dostupné z URL: <http://blinkdagger.com/matlab/matlab-gui-graphical-user-interface-tutorial-for-beginners/>.

[10] MATLAB Central. *MATLAB Central: An open exchange for the MATLAB and Simulink user community* [online]. [cit. 5. 5. 2010]. Dostupné z URL: <http://www.mathworks.com/matlabcentral/newsreader/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

PSC Parametr Scheduling Control

T_s vzorkovací perioda

GUI grafické uživatelské rozhraní – Graphical user interface

K zesílení regulátoru

T_I integrační konstanta regulátoru

T_D derivační konstanta regulátoru

N konstanta omezující zesílení pro vyšší frekvence regulátoru

F_S přenos soustavy

F_R přenos regulátoru

F_W přenos řízení

GUIDE Graphical User Interface Development Environment