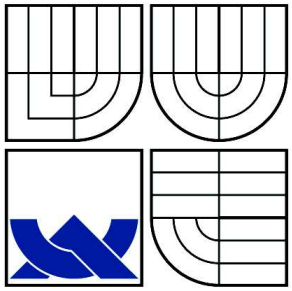
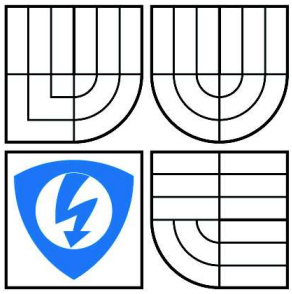


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

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

PRŮMYSLOVÝ PID REGULÁTOR S VIZUALIZACÍ A BEZNÁRAZOVÝM PŘEPÍNÁNÍM

INDUSTRIAL PID CONTROLLER WITH VISUALIZATION AND BUMPLESS SWITCHING

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

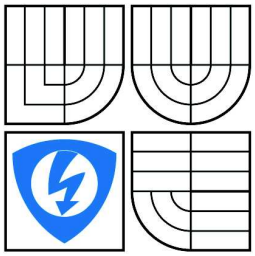
AUTOR PRÁCE — MICHAL KUPČÍK

AUTHOR

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

SUPERVISOR

BRNO 2009



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:	Michal Kupčík	***ID:***	72939
Ročník:	3	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Průmyslový PID regulátor s vizualizací a beznárazovým přepínáním

POKYNY PRO VYPRACOVÁNÍ:

Vytvořte průmyslově použitelný regulační člen s diskrétním PID regulátorem s antiwindupem a beznárazovým přepínáním. Implementujte jej do zařízení Power Panel firmy B&R, které integruje programovatelný automat a operátorský panel s dotykovou obrazovkou. Regulační člen bude ovládán přes dotykovou obrazovku a bude umožňovat změnu struktury regulátoru, změnu parametrů regulátoru a možnost přepnutí na ruční režim ovládání.

DOPORUČENÁ LITERATURA:

PIVOŇKA, P.: Číslicová řídicí technika, VUT Brno, skriptum, 2003

Termín zadání:	9.2.2009	***Termín odevzdání:***	1.6.2009

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

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

UPOZORNĚNÍ:

ANOTACE

Tato práce seznámí čtenáře s různými programovými realizacemi PID regulátoru. Dále obsahuje popis čtyř základních událostí, u kterých dochází k nárazovému přepínání. Pro každou z těchto čtyř událostí je provedena analýza, proč dochází k nárazu a následně i popis, jak tyto nárazy eliminovat. Výsledkem je sada rovnic, které vhodně upravují parametry regulátorů. Na velkém počtu grafů je dokázáno, že po aplikaci těchto rovnic již nadále k nárazům nedochází. Další součástí dokumentu je popis, jak názorně ukázat beznárazové přepínání ve skriptu MATLABu, ve virtuálním řídicím systému AR000 firmy B&R a na reálné soustavě, která bude řízená řídicím systémem typu Power Panel firmy B&R.

KLÍČOVÁ SLOVA

Beznárazové přepínání, PID regulátor, Power Panel

ANNOTATION

This work informs reader about different program implementation of PID controller. Next it contains description of four basic events, which cause bump switching. For each of these four events is made analysis, why bump is caused and subsequently is made description, how to eliminate these bumps. The result is set of equations, which properly adjust regulators parameters. On big number of graphs there is proved, that after application of these equations, switching become bumpless. Another part of this dokument is a description, how to illustrative demonstrate bumpless switching in MATLAB's script, in B&R's virtual control system AR000 and on real system, whitch will be controled by B&R's control system type Power Panel.

KEYWORDS

Bumpless switching, PID controller, Power Panel

Bibliografická citace

KUPČÍK, M. *Průmyslový PID regulátor s vizualizací a beznárazovým přepínáním*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 64 s., 4 přílohy. Vedoucí bakalářské práce prof. Ing. Petr Pivoňka, CSc.

P r o h l á š e n í

„Prohlašuji, že svou bakalářskou práci na téma „Průmyslový PID regulátor s vizualizací a beznárazovým přepínáním“ 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:

P o ď ě k o v á n í

Na tomto místě bych chtěl poděkovat všem doktorandům vyučující počítačové cvičení z předmětu BCRT, jmenovitě Ing. Luďkovi Chomátovi, Ing. Petrovi Malounkovi a Ing. Martinovi Dvořáčkovi za jejich pomoc při tvorbě programu v sw B&R.

Rovněž bych chtěl poděkovat svému vedoucímu semestrální práce prof. Ing. Petrovi Pivoňkovi CSc. za cenné rady a připomínky.

Dále bych chtěl poděkovat svým příbuzným a známým za všestrannou podporu po celou dobu mého studia.

V Brně dne : Podpis:

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1. ÚVOD

Každý průmyslově použitelný regulátor by měl mít v sobě implementovány algoritmy, které umožní změnit režim regulátoru z automatického na manuální a naopak, aniž by se jakkoliv změnila velikost akčního zásahu. V opačném případě by došlo k jevu v praxi označovanému jako „nárazu“ akčního zásahu. Toto by se mohlo projevit až destrukcí části regulované soustavy.

Regulujeme-li t-variantní systém, jehož vlastnosti se časem mění, regulátorem, který průběžně sleduje změny vlastností soustavy a podle nich mění své konstanty, musíme v programu regulátoru zajistit, aby i tato změna konstant proběhla beze změny velikosti akčního zásahu.

Pokud bylo při návrhu regulátoru zjištěno, že při stejných parametrech regulátoru mají různě varianty algoritmu výrazně lepší vlastnosti v různých situacích, například při změně žádané hodnoty nebo při poruše, bude jistě záhodno mezi těmito algoritmy dynamicky přepínat v průběhu regulace. Rovněž toto přepínání musí být realizováno tak, aby se velikost akčního zásahu nezměnila.

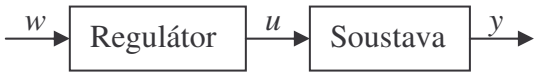
Tato práce má za úkol nalézt právě tyto algoritmy, po jejichž aplikaci v programu regulátoru se při výše zmiňovaných událostech hodnota akčního zásahu nezmění a přepnutí proběhne beznárazově a následně výsledky práce prezentovat ve formě spustitelného skriptu MATLABu a programu v řídicím systému B&R, jenž bude regulovat reálnou soustavu.

2. ZÁKLADNÍ POJMY

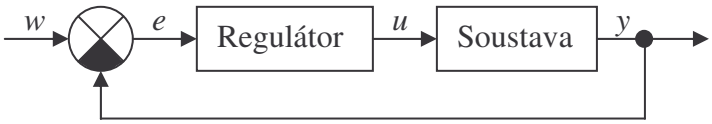
2.1 REGULAČNÍ OBVOD

Odlišují se dva základní typy regulačního obvodu: uzavřený a otevřený. U otevřeného regulačního obvodu nemá regulátor informaci o aktuálním stavu systému a jeho okamžitém výstupu a proto se tento typ regulace označuje jako ovládání.

V uzavřeném obvodu má díky zpětné vazbě regulátor informaci o aktuálním výstupu z regulované soustavy *y*. Regulátor nyní nereaguje přímo na žádanou hodnotu *w*, ale na regulační odchylku *e*, jež je rozdílem mezi žádanou hodnotou a výstupní hodnotou ze soustavy. Výstupem ze soustavy je akční zásah *u*.



Obr. 2.1: Otevřený regulační obvod.



Obr. 2.2: Uzavřený regulační obvod.

2.2 REGULÁTOR

Regulátor si můžeme představit jako černou krabičku, o které nevíme, jak uvnitř funguje, ale pro nás je důležité, že snahou regulátoru je měnit svou hodnotu na výstupu tak, aby se jeho vstupní hodnota rovnala nule, nebo se jí co nejvíce blížila.

Ve své době se v této krabičce nalézaly analogové součástky jako operační zesilovače, kondenzátory, odpory a jiné. V dnešní době většinu těchto součástek nahradil jeden programovatelný jednočip a na něj připojené vstupně výstupní periferie.

Jakým způsobem tento jednočip naprogramujeme, záleží již pouze na uživateli. V našem případě do něj nahrajeme algoritmy popisující chování regulátorů dle [1] a to jak paralelní tvar

$$F_{R1}(s) = K\left(1 + \frac{1}{T_I s} + \frac{T_D s}{\frac{T_D}{N} s + 1}\right) \tag{2.1}$$

, který budeme nazývat PID regulátorem a z něj odvozené varianty PI-D a I-PD, tak sériový tvar

$$F_{R2}(s) = K\left(1 + \frac{1}{T_I s}\right)\left(\frac{T_D s + 1}{\alpha T_D s + 1}\right) \quad ; \quad \alpha = 0{,}1 \tag{2.2}$$

, jenž v této práci budeme označovat jako PI*D a I-P*D regulátory.

Třetí tvar regulátoru, pro který použijeme název PSD regulátor, je popsán rovnicí

$$F_{R3}(z) = \frac{a + bz^{-1} + cz^{-2}}{1 - z^{-1}} \tag{2.3}$$

Protože řídicí systémy neumí pracovat se spojitou verzí rovnice (2.1) a (2.2), převedeme si je na diskrétní ekvivalenty

$$F_{R1}(z) = K\left(1 + \frac{T_{VZ}}{T_I}\frac{z^{-1}}{1 - z^{-1}} + N\frac{1 - z^{-1}}{1 - e^{-\frac{T_{VZ}\cdot N}{T_D}} z^{-1}}\right) \tag{2.4}$$

a

$$F_{R2}(z) = K\left(1 + \frac{T_{VZ}}{T_I}\frac{z^{-1}}{1 - z^{-1}}\right)\left(10\frac{1 - z^{-1}}{1 - e^{-\frac{T_{VZ}}{0{,}1T_D}} z^{-1}} + \frac{\left(1 - e^{-\frac{T_{VZ}}{0{,}1T_D}}\right) z^{-1}}{1 - e^{-\frac{T_{VZ}}{0{,}1T_D}} z^{-1}}\right) \tag{2.5}$$

Ani tyto rovnice v tomto znění do PLC nenaprogramujeme a tak provedeme závěrečnou úpravu například rovnice (2.3) na tvar

$$\frac{U(z)}{E(z)} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 - z^{-1}} \;\rightarrow\; U(z)\left(1 - z^{-1}\right) = E(z)\left(a_0 + a_1 z^{-1} + a_2 z^{-2}\right) \;\rightarrow$$

$$Z^{-1} \Rightarrow u(k) = ae(k) + be(k-1) + ce(k-2) + u(k-1) \tag{2.6}$$

Předcházející úprava je více rozebrána v následující podkapitole. Zpoždění vzorků již provést umíme, stejně tak i součet.

Více o jednotlivých regulátorech bude napsáno v následující kapitole.

2.3 SOUSTAVA

Ve skriptu MATLABu a ve virtuálním řídicím systému B&R budeme používat pro ukázky soustavu ve spojitém tvaru

$$F_S(p) = \frac{1}{(3p+1)(2p+1)} \tag{2.7}$$

Chceme-li simulovat soustavu programem, musíme znát přenos soustavy v diskrétním tvaru. Jako vzorkovací periodu si zvolíme T_{VZ} = 0,1 s. Pro převod ze spojitého tvaru na tvar diskrétní použijeme příkaz MATLABu *c2d()*. Obdržíme následující rovnici

$$F_S(z) = \frac{0{,}0008105z^{-1} + 0{,}0007883z^{-2}}{1 - 1{,}918z^{-1} + 0{,}92z^{-2}} \tag{2.8}$$

kterou následně upravíme na

$$\frac{Y(z)}{U(z)} = \frac{0{,}0008105z^{-1} + 0{,}0007883z^{-2}}{1 - 1{,}918z^{-1} + 0{,}92z^{-2}} \Rightarrow$$
$$\Rightarrow Y(z)\left(1 - 1{,}918z^{-1} + 0{,}92z^{-2}\right) = U(z)\left(0{,}0008105z^{-1} + 0{,}0007883z^{-2}\right) \tag{2.9}$$

a provedeme zpětnou Z-transformaci

$$\begin{aligned} &y(k) - 1{,}918y(k-1) + 0{,}92y(k-2) = \\ &= 0{,}0008105u(k-1) + 0{,}0007883u(k-2) \end{aligned} \tag{2.10}$$

poslední úpravou osamostatníme *y(k)*

$$\begin{aligned} &y(k) = 0{,}0008105u(k-1) + 0{,}0007883u(k-2) - \\ &-\left(-1{,}918y(k-1) + 0{,}92y(k-2)\right) \end{aligned} \tag{2.11}$$

Pokud si označíme $Y = y(k)$, $Y1 = y(k-1)$, $Y2 = y(k-2)$, $U = u(k)$, $U1 = u(k-1)$a $U2 = y(k-2)$, potom nebude problém naprogramovat chování soustavy pomocí jedné výpočtové rovnice a série zpožďovacích přiřazovacích rovnic.

Když se ale podíváme na skutečná čísla v MATLABu u Y1 a Y2, zjistíme, že mají mnoho desetinných míst. Kdybychom už tyto hodnoty u U1, U2, Y1 a Y2 nechali tak jak je nyní máme, dostali bychom soustavu s jistým zesílením rozdílným

od jedné. Proto si k hodnotě u Y1 dopíšeme další 4 desetinná místa tak, abychom dostali zesílení 1: -1,9184012.

```
Y = 0,0008105 * U1 + 0,0007883 * U2 -
    ( - 1,9184012 * Y1 + 0,92 * Y2);
U2 = U1;
U1 = U;
Y2 = Y1;
Y1 = Y;
```

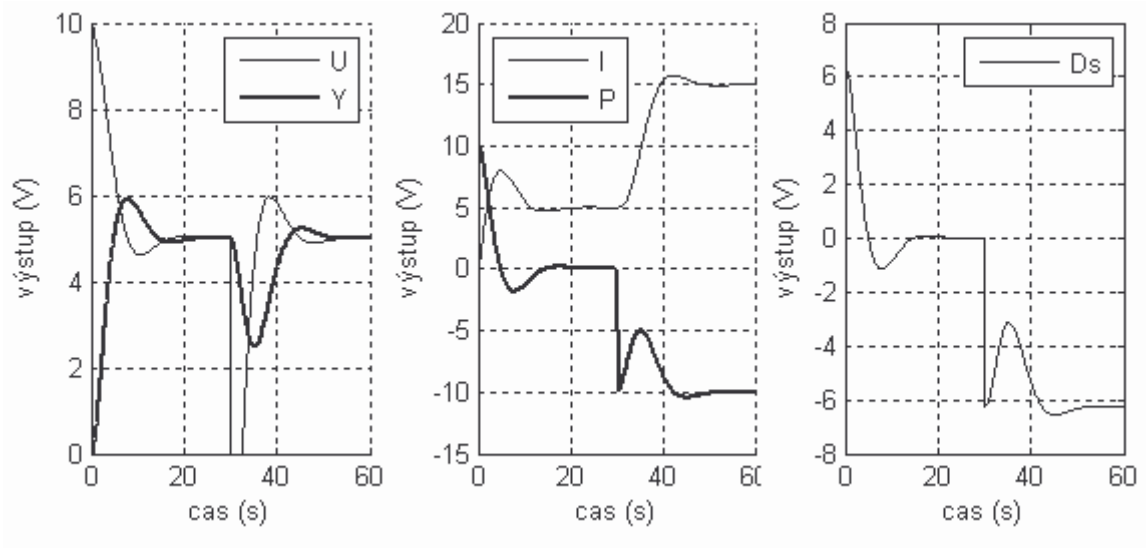
2.4 BEZNÁRAZOVÉ PŘEPÍNÁNÍ

Vyřešení problému beznárazového přepínání je jeden z nutných kroků před zavedením regulátoru do praxe. Nárazem se označuje velká skoková změna akčního zásahu při jednom z následujících čtyř úkonů při nevyřešeném beznárazovém přepínání:

1. Přepínání mezi algoritmy regulátoru jednoho typu.
2. Přepínání mezi rozdílnými typy regulátorů.
3. Přepínání mezi automatickým a manuálním režimem.
4. Změna hodnoty vnitřní konstanty regulátoru (K, T_D, N) na jinou.

Manuálním režimem se označuje stav, kdy velikost akčního zásahu neovlivňuje regulátor ale operátor.

Jako názornou ukázku nárazu vybereme přepnutí z PID regulátoru na I-PD algoritmus v době po ustálení přechodového děje. V čase t = 30s dojde ke změně řídicího algoritmu a k nárazu, viz obr. 2.3.



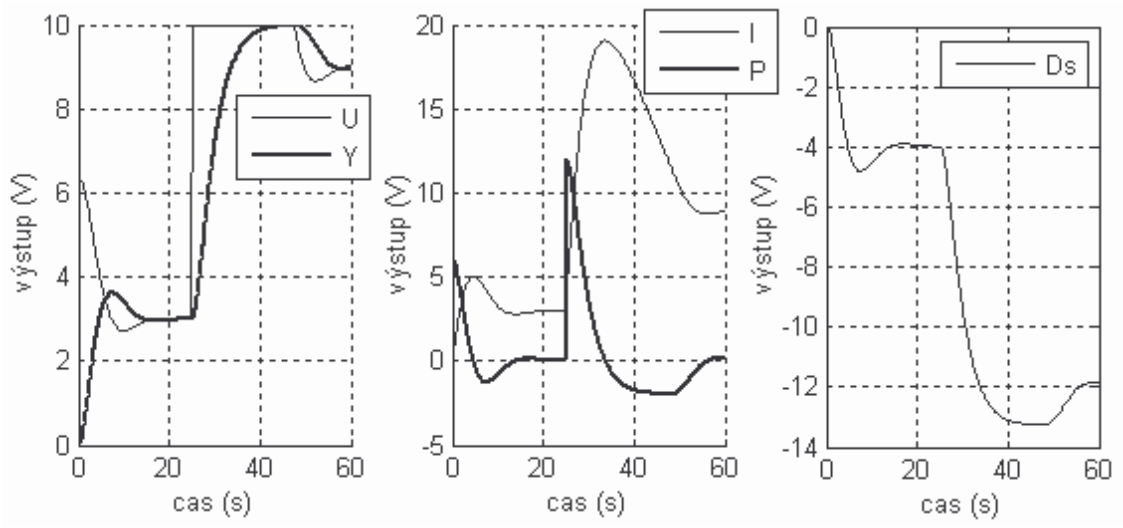
Obr. 2.3: Nárazové přepnutí z PID na I-PD

2.5 ANTIWINDUP

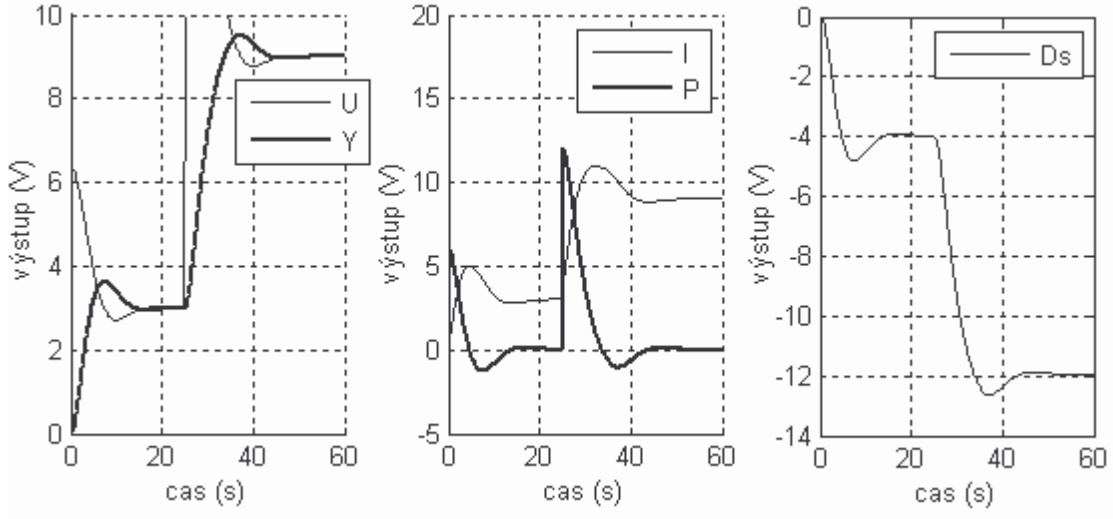
Pomocí Antiwindupu se omezuje další nárůst hodnoty integrační (sumační) složky regulátoru, pokud již bylo dosaženo saturace akčního zásahu a ten se již nemůže dále zvětšovat. V opačném případě by došlo k prodloužení regulačního děje o tu dobu, po kterou by musel integrátor svou hodnotu opět odečítat. Ačkoliv PSD regulátor žádnou sumační složku nemá, toto omezení využije také, neboť do výpočtu aktuálního akčního zásahu se připočítává minulý akční zásah a pokud ten by stále rostl, obdrželi bychom stejný jev přebuzení jako u PID regulátoru.

Antiwindup rovněž nazýváme dynamickým omezením sumační složky a v programu jej zapíšeme kódem

```
if U>akc_zas
 Suma=Suma+(akc_zas-U)*K*Tvz/Tt;
 U=akc_zas;
End
```



Obr. 2.4: Projevení windupu.



Obr. 2.5: Ošetření windupu.

Na předcházejících dvou obrázcích vidíme zrychlení regulačního děje. V prvním případě hodnota sumační složky neúměrně naroste nad hodnotu 18, ale díky antiwindupu nepřesáhne hodnoty 12. Žádaná hodnota je 9 a je tedy jasné, že z hodnoty 12 se na tuto hodnotu dostane sumační složka podstatně rychleji, než z hodnoty 18.

3. REGULÁTORY

3.1 PID, PI-D, I-PD

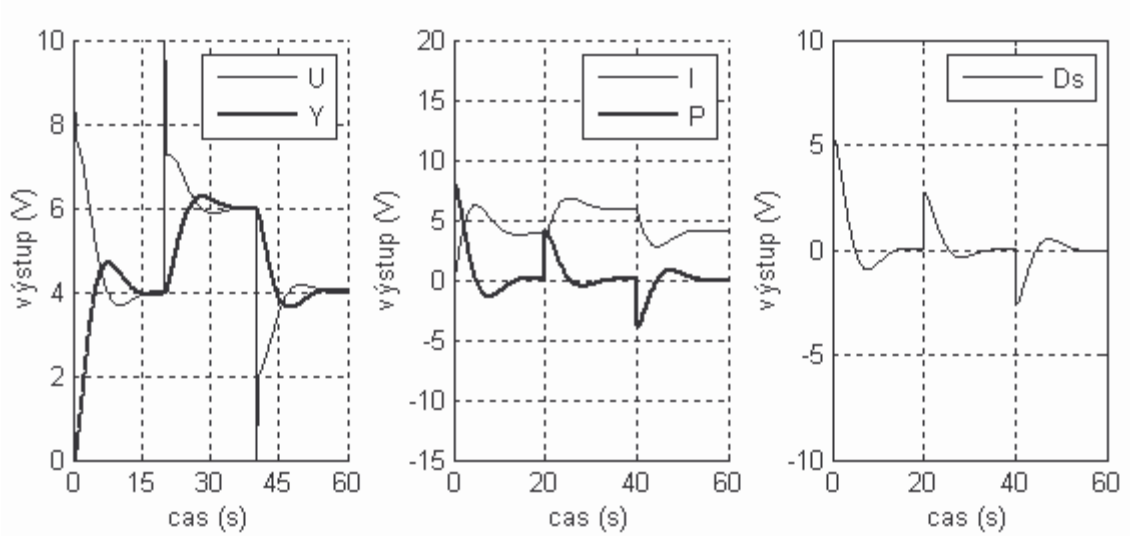
a jeho obměny PI-D a I-PD patří mezi nejvíce rozšířené typy regulátorů. Přenosová rovnice PID regulátoru v Z-transformaci je uvedená v (2.4):

$$F_R(z) = \frac{U(z)}{E(z)} = K\left(1 + \frac{T_{VZ}}{T_I}\frac{z^{-1}}{1-z^{-1}} + N\frac{1-z^{-1}}{1-e^{-\frac{T_{VZ}\cdot N}{T_D}}z^{-1}}\right)$$

Vidíme, že přenos PID regulátoru je dán součtem přenosů proporcionální, integrační a derivační části. Proporcionální složka rovnou zesiluje regulační odchylku $e = w - y$. Do výstupu z integrační složky se započítává jeho předcházející hodnota a o jeden krok vhodně vynásobená zpožděná hodnota regulační odchylky. Derivační složka zajišťuje zesilování změn na jeho vstupu.

Algoritmus PID regulátoru vypadá následovně:

```
Der=N*K*((W-Y)-Ds+Ds*exp(-Tvz*N/Td));
Prop=K*(W-Y);
U=Prop+Suma+Der;
Ds=(W-Y)+Ds*exp(-Tvz*N/Td);
Suma=Suma+K*Tvz/Ti*(W-Y);
```



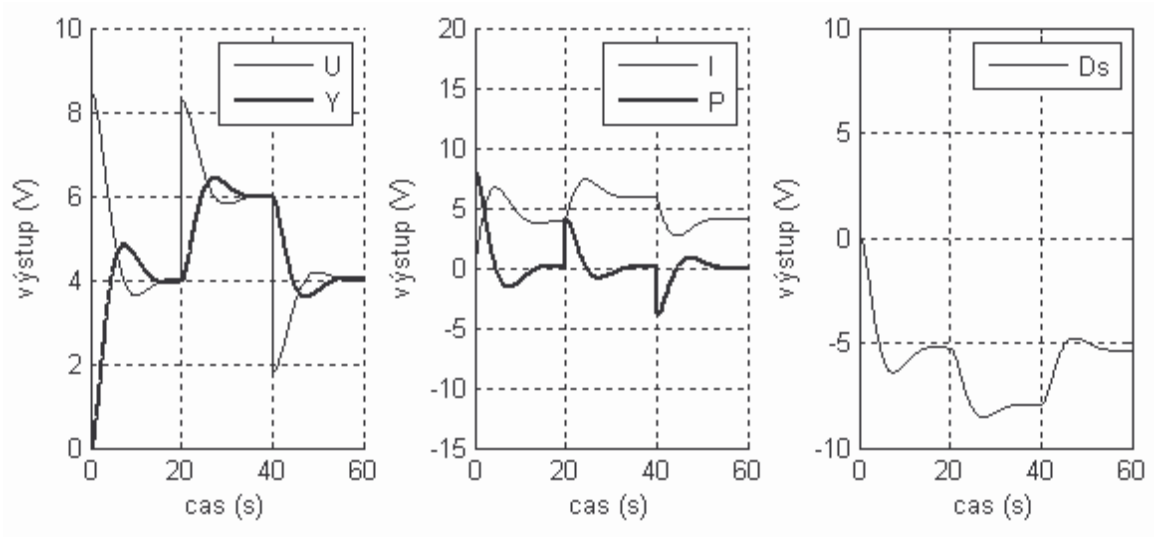
Obr. 3.1: Časová odezva PID regulátoru a soustavy.

V čase 0, 20 a 40 sekund vidíme špičky akčního zásahu způsobené derivační složkou, která reaguje na změny žádané hodnoty. Tyto špičky mohou být nežádoucí

a zbavíme se jich jednoduchou změnou vstupní hodnoty. Místo regulační odchylky *e* bude jako vstup sloužit záporná hodnota výstupu ze soustavy *–y*. V algoritmu regulátoru to bude znamenat změnu v prvním a čtvrtém řádku na:

```
Der=N*K*((-Y)-Ds+Ds*exp(-Tvz*N/Td));
Ds=(-Y)+Ds*exp(-Tvz*N/Td);
```

Tento algoritmus pojmenujeme PI-D.



Obr. 3.2: Časová odezva PI-D regulátoru a soustavy.

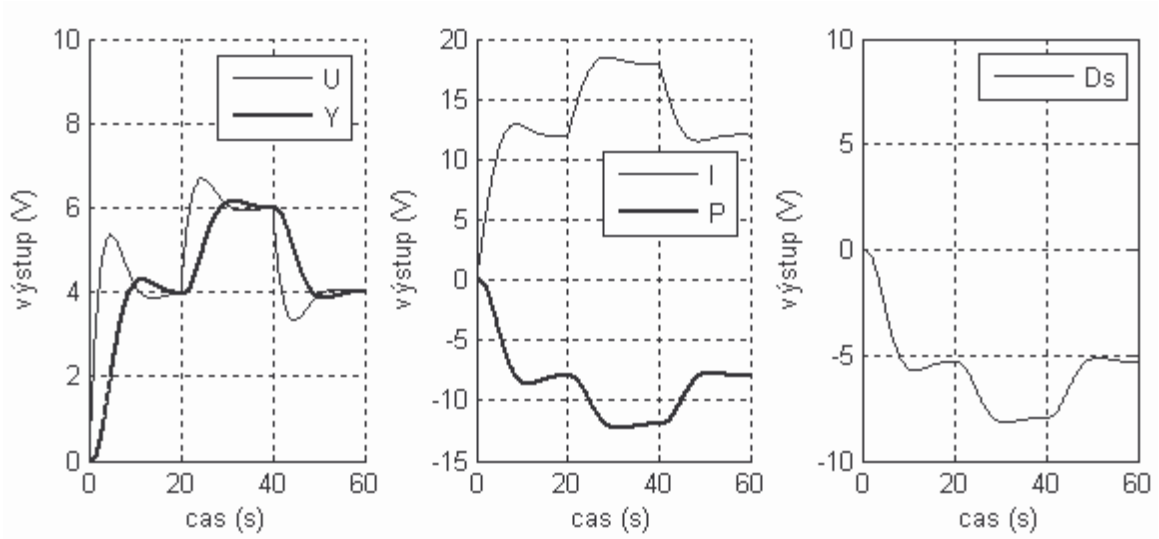
Abychom mohli volně přepínat mezi těmito dvěma algoritmy, použijeme proměnnou *Alfa*. Ta nám bude řídit přítomnost žádané hodnoty *w* na vstupu derivační složky:

```
Der=N*K*((Alfa*W-Y)-Ds+Ds*exp(-Tvz*N/Td));
Ds=(Alfa*W-Y)+Ds*exp(-Tvz*N/Td);
```

Pokud provedeme podobnou úpravu i u proporcionální složky, nyní s proměnnou *Beta*, obdržíme regulátor I-PD při *Beta* = 0.

Finální tvar algoritmu obsluhující regulátoru PID, PI-D a I-PD vypadá:

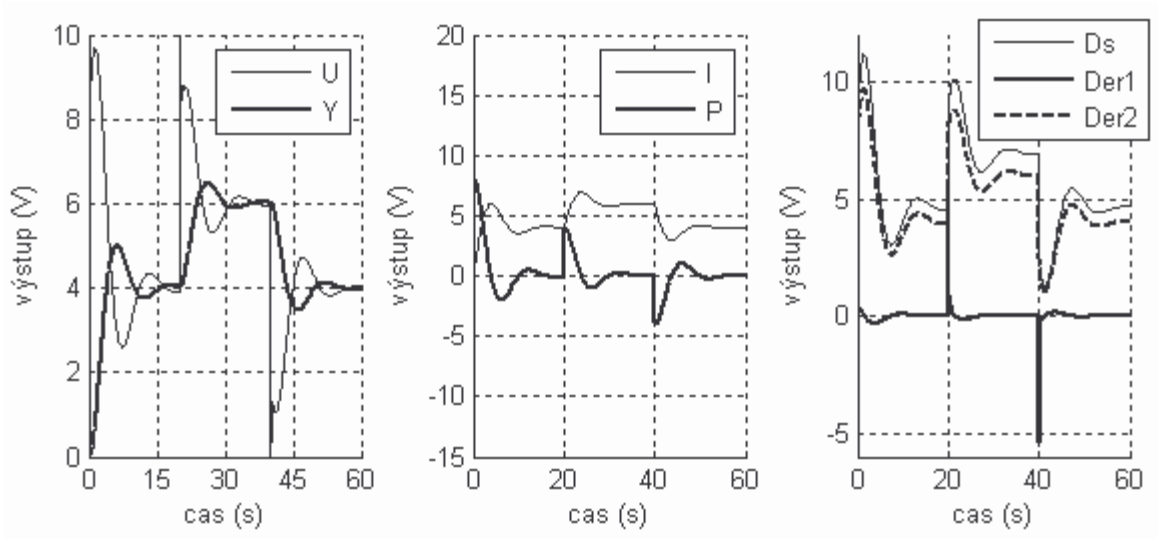
```
Der=N*K*((Alfa*W-Y)-Ds+Ds*exp(-Tvz*N/Td));
Prop=K*(Beta*W-Y);
U=Prop+Suma+Der;
Ds=(Alfa*W-Y)+Ds*exp(-Tvz*N/Td);
Suma=Suma+K*Tvz/Ti*(W-Y);
```



Obr. 3.3: Časová odezva I-PD regulátoru a soustavy.

3.2 PI*D, I-P*D

Rovnici, popisující chování PID regulátoru v sériovém tvaru, jsme si již odvodili v (2.5). Protože máme zvlášť rozdělenou PI a D část, označíme si tento regulátor jako PI*D. Kromě derivační části vidíme v druhé závorce i setrvačný článek 1. řádu, který zajistí, aby se hodnota akčního zásahu vypočítaná v PI části postupem času přenesl na výstup celého regulátoru.

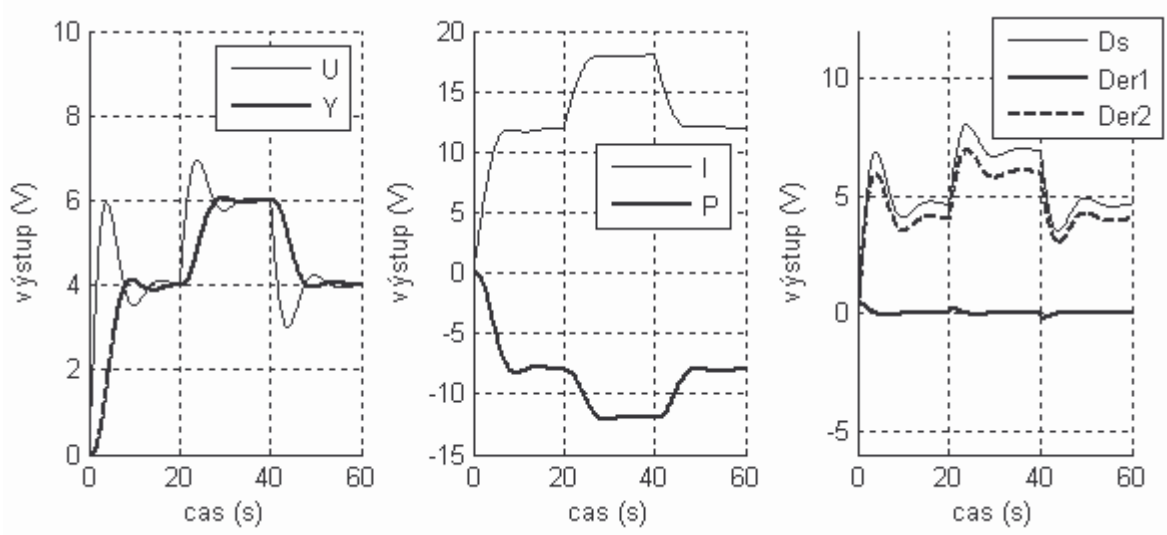


Obr. 3.4: Časová odezva PI*D regulátoru a soustavy.

Algoritmus, který toto reprezentuje vypadá následovně:

```
Der1=10*(Upi-Ds+Ds*exp(-Tvz*10/Td));
Prop=K*(Beta*W-Y);
Upi=Prop+Suma;
Ds=(Upi)+Ds*exp(-Tvz*10/Td);
Suma=Suma+K*Tvz/Ti*(W-Y);

Der2=(1-exp(-Tvz/(Td/10)))*R_E1+(exp(-Tvz/(Td/10)))*R_U1;
R_E1=Upi;
R_U1=Der2;

U=Der1+Der2;
```

U tohoto algoritmu vidíme u proporcionální části stejnou úpravu, jakou jsme provedli u PID regulátoru. Změnou hodnoty proměnné *Beta* můžeme řídit, zda má proporcionální složka reagovat na změny regulační odchylky *e* a tím vytvářet krátké špičky hodnoty akčního zásahu, nebo ne a díky toho získáme průběh regulace podobný jako u I-PD regulátoru, který označíme jako I-P*D regulátor.



Obr. 3.5: Časová odezva I-P*D regulátoru a soustavy.

3.3 PSD, S-PD

Přenos PSD regulátoru jsme si odvodili v rovnici (2.6) jako

$$u(k) = ae(k) + be(k-1) + ce(k-2) + u(k-1)$$

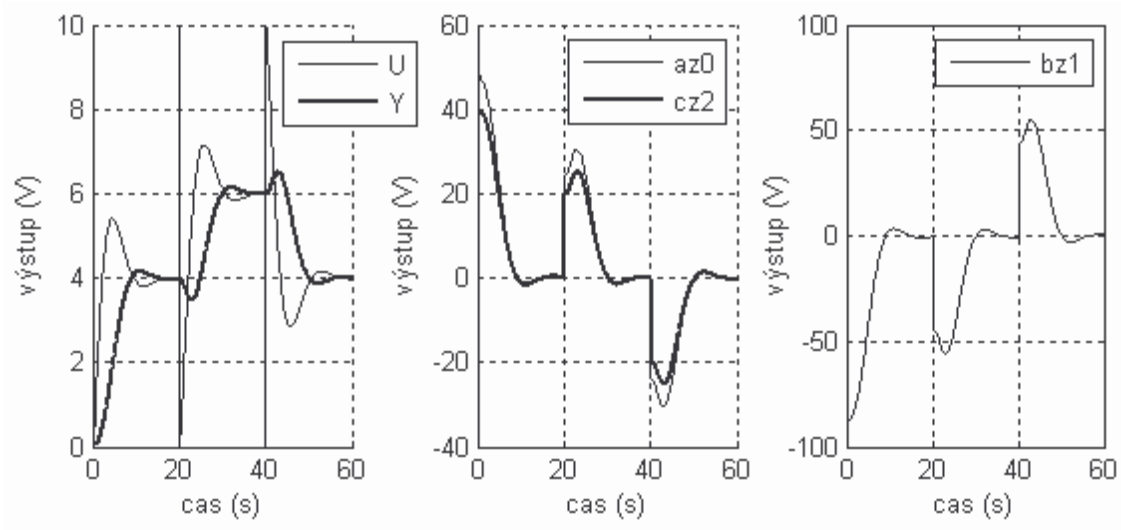
Nejdříve si vypočítáme jednotlivé konstanty *a*, *b* a *c* jako

```
a=K*(1+Tvz/Ti+Td/Tvz);
b=-K*(1+2*Td/Tvz);
c=K*Td/Tvz;
```

Poté sečteme jednotlivé členy a provedeme zpoždění vzorků o jeden krok.

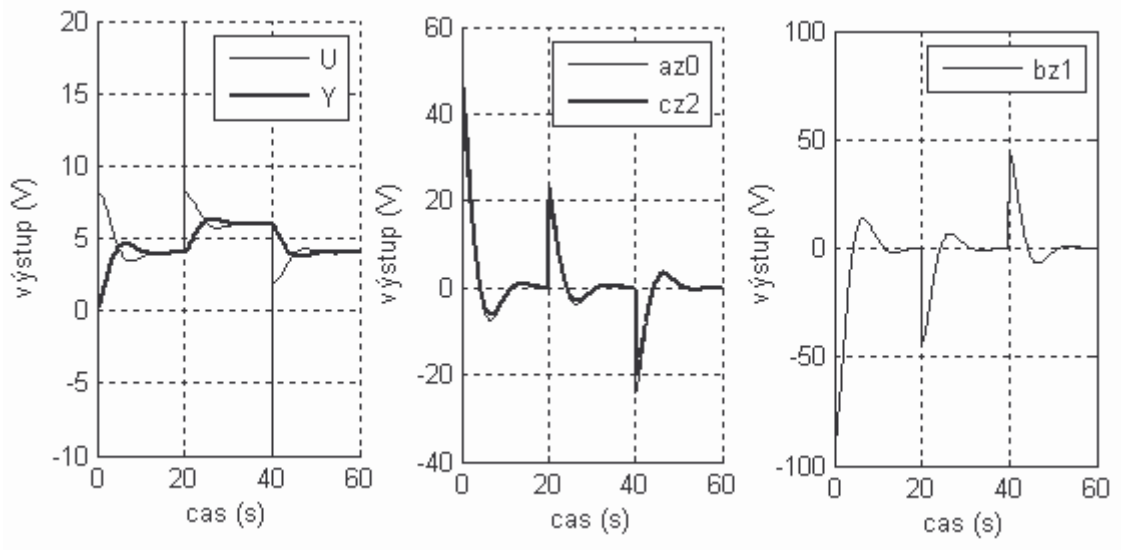
```
R_E0=W-Y;
U=a*R_E0+b*R_E1+c*R_E2+R_U1;
R_E2=R_E1;
R_E1=R_E0;
```

Zpoždění `R_U1 = U` provedeme až za antiwindup rovnicemi.



Obr. 3.6: Průběh regulace PSD algoritmem s antiwindup rovnicemi.

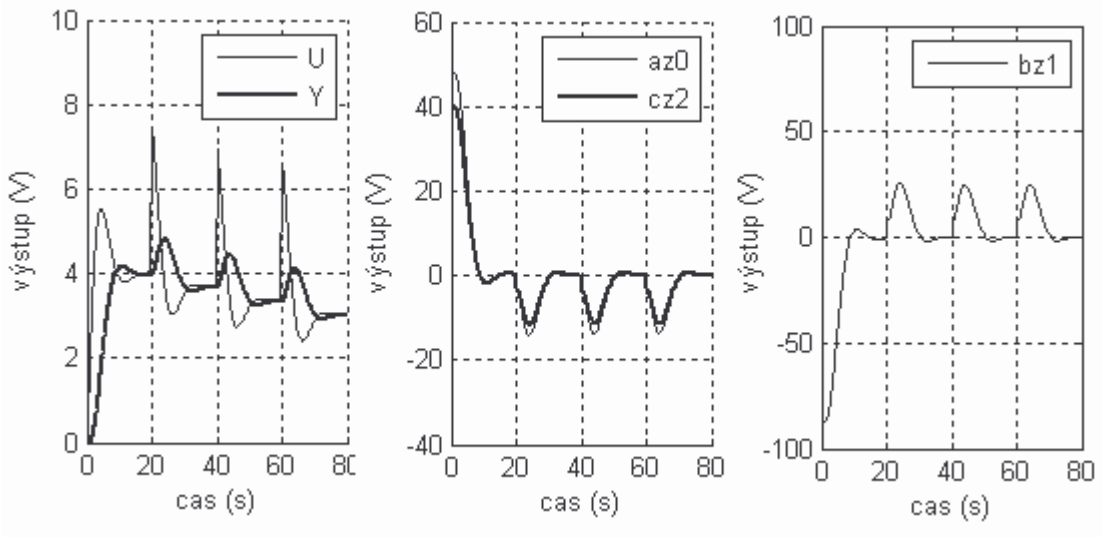
Z předchozího obrázku vidíme, že antiwindup rovnice tak, jak je máme nyní sestavené, způsobí zákmity akčního zásahu, díky kterých soustava zareaguje změnou své výstupní hodnoty přesně na tu druhou stranu, než je žádáno.



Obr. 3.7: Průběh regulace PSD algoritmem bez antiwindup rovnic.

Rozhodneme se tedy programově odstranit tyto zákmity akční veličiny. První možnost, která se nabídne je pamatování si předchozího akčního zásahu a pokud nově vypočtený akční zásah je větší jak maximální akční zásah nebo menší jak minimální, tak místo nově spočítaného akčního zásahu použijeme předchozí.

Toto řešení však zklame, pokud budeme měnit žádanou hodnotu v malých krocích, neboť druhý nově vypočtený akční zásah bude stále v mezích.



Obr. 3.8: Problém změny žádané hodnoty ze 4 na 3 ve třech krocích u PSD.

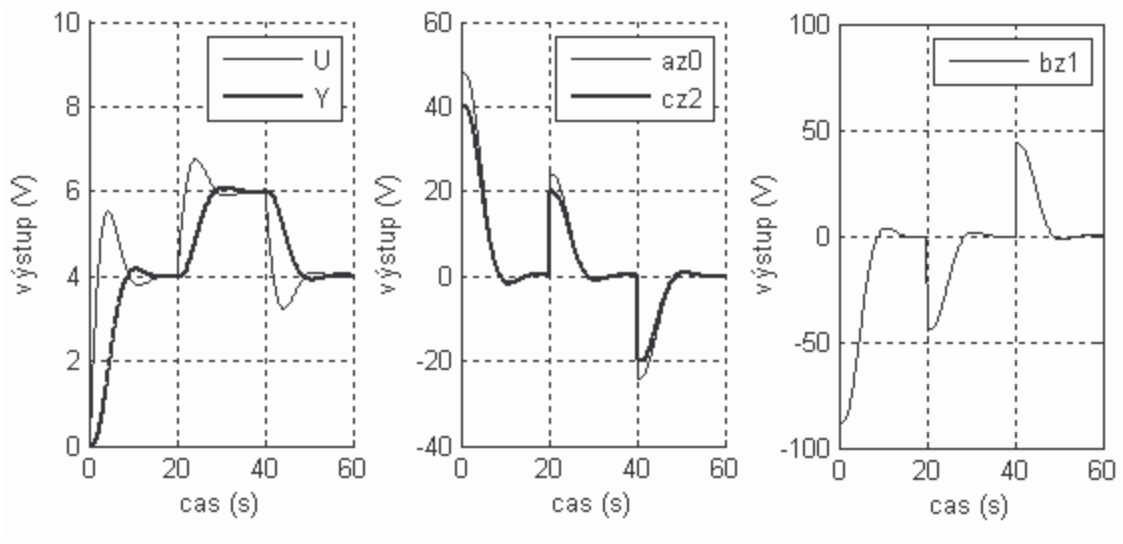
Další algoritmus který vyzkoušíme je založen na porovnávání dvou po sobě jdoucích akčních zásahů. Pokud je tedy současný akční zásah vetší či menší o nějakou danou mez než v předchozím kroku, místo spočteného akčního zásahu vložíme na výstup regulátoru akční zásah předchozí. Pokud ale provedeme dostatečně malý skok žádané hodnoty, tyto špičky opět objevíme.

Poslední způsob který vyzkoušíme je porovnávání dvou po sobě jdoucích žádaných hodnotách. Tyto překmity a zákmity totiž vznikají pouze při změně žádané hodnoty. Zaznamenáme-li tedy změnu žádané hodnoty, dva následující vypočtené akční zásahy nahradíme akčním zásahem, který regulátor vypočetl před změnou žádané hodnoty. Před rovnice regulátoru vložíme kód

```
if Wpred ~= W      %pokud W neni totozne jako Wpred
    WrozdilCount=2;
end
```

A za rovnice regulátoru kód

```
if WrozdilCount>0
    U=Upred;
    WrozdilCount=WrozdilCount-1;
end
```



Obr. 3.9: Průběh PSD regulátoru s S-PD ošetřením špiček.

Průběh na předchozím obrázku se nápadně podobá průběhu na obr. 3.3 u I-PD algoritmu. Proto tuto úpravu zavedeme označení S-PD regulátor.

Nyní vymyslíme jiný způsob, jak zamezit zákmitům výstupní hodnoty na druhou stranu než je žádáno z obr. 3.6. Jádrem myšlenky je uložení si bokem hodnotu, která přesahuje meze akčního zásahu a tuto hodnotu následně přičíst v rovnici výpočtu akčního zásahu. Tuto hodnotu je ale potřeba udržovat pouze dočasně po dobu maximálně 2 vzorků, neboť stálé přičítání rozdílové hodnoty k vypočtenému akčnímu zásahu by neúměrně prodloužilo dobu regulace.

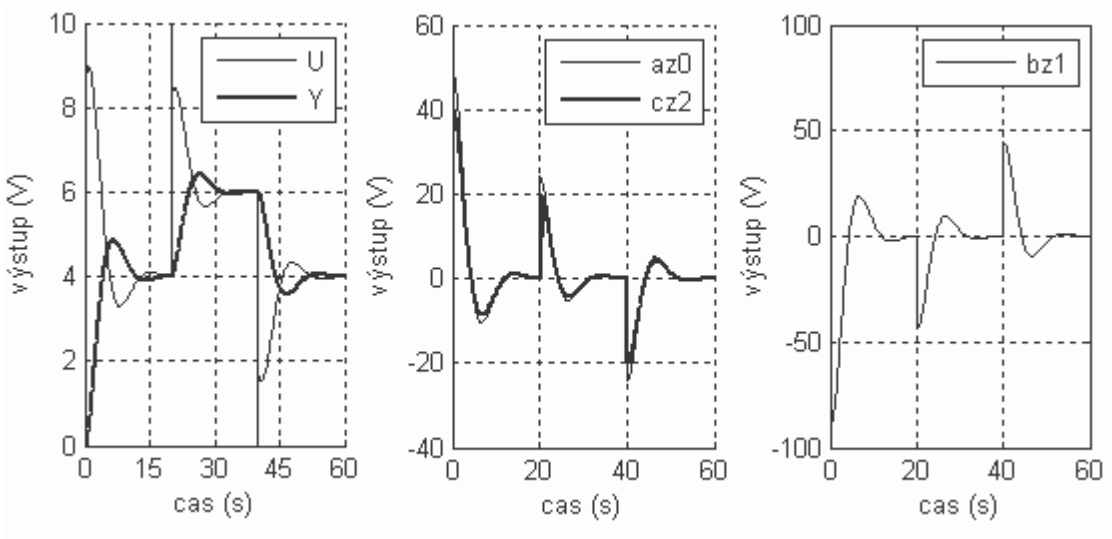
Do rovnic antiwindupu tedy doplníme kód

```
if UrozdilCount<3
    Urozdil=U-akc_zas;
    UrozdilCount=UrozdilCount+1;
else
    Urozdil=0;
end
```

a to pro obě podmínky. Rovnici pro výpočet akčního zásahu upravíme do následujícího znění:

```
U=az0+bz1+cz2+R_U1+Urozdil;
```

Toto ošetření špiček budeme nazývat PSD regulátorem.



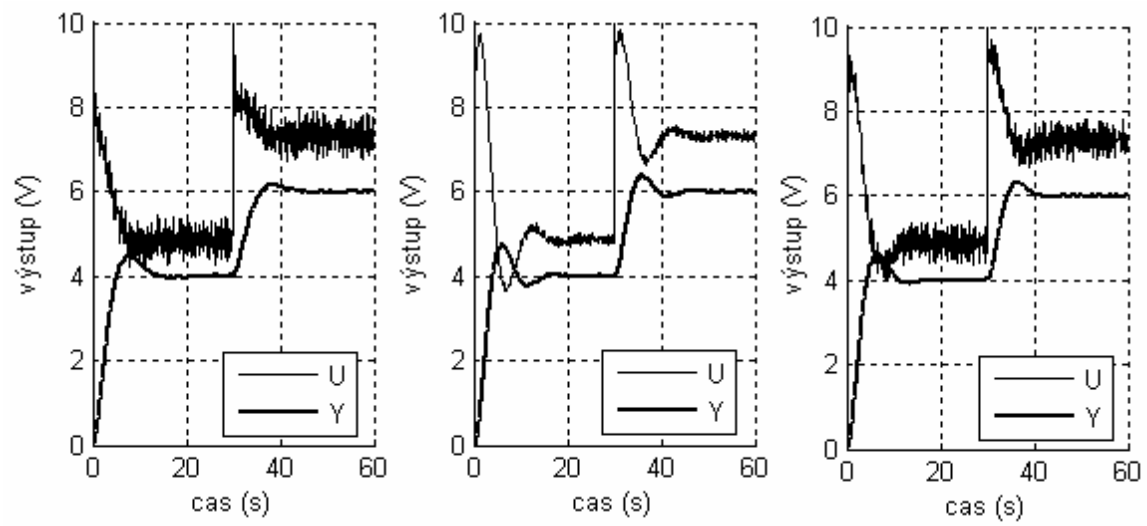
Obr. 3.10: Časový průběh PSD regulátoru a soustavy.

Zároveň doplníme algoritmus o proměnnou *TypPSD*, která určí, jakým způsobem se budou zpracovávat špičky akčního zásahu při změně žádané hodnoty. Budeme tak vybírat mezi PSD a S-PD regulátorem.

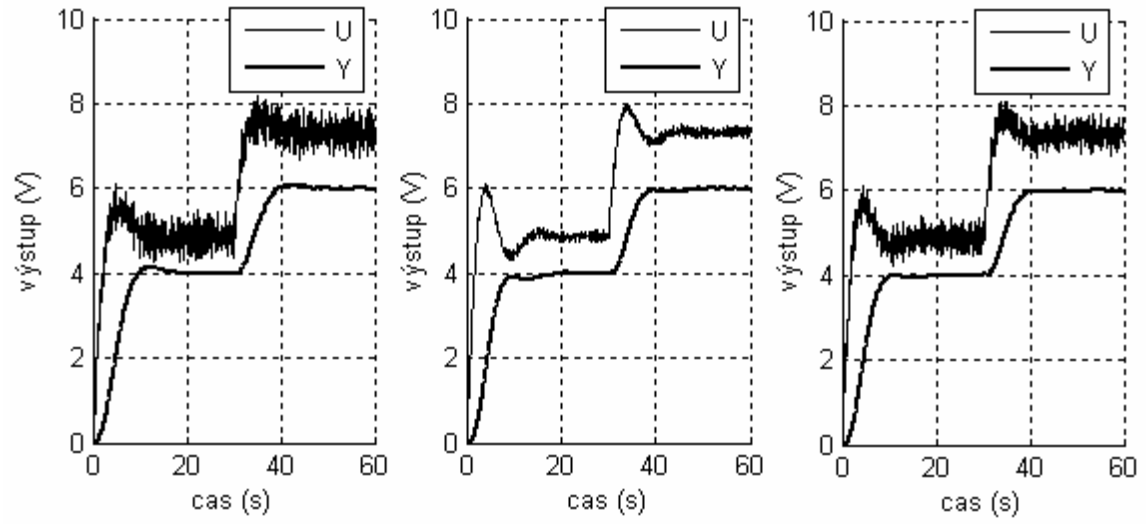
3.4 PŘENOS ŠUMŮ

Při regulaci v reálném prostředí s reálnou soustavou vznikají v obvodu šumy, které mohou podstatně ovlivnit správnou funkci regulátoru, neboť derivační složka v regulátoru zesiluje vysoké frekvence. Proto musíme tento vliv šumů filtrovat. U PID, PI-D a I-PD regulátoru je tato filtrace zajištěna filtračním koeficientem N, který můžeme běžně nastavit v rozsahu 3 (nejvíce filtrované) až po 20. U PI*D a I-P*D regulátoru by měla být hodnota fixně nastavena na $N = 10$, neboť $\alpha = 0{,}1 = 1/N$. U PID alternativ regulátorů použitých v této práci používáme $N = 7$. Toto číslo je menší než u PI*D regulátoru, přenos šumů by tedy měl být u PID variant menší. Ovšem jak uvidíme na následujícím obrázku, realita je jiná. U PSD regulátoru žádná proměnná N není.

Na následujících dvou obrázcích není použitá soustava s přenosem 1, aby bylo možné sledovat zvlášť průběh akčního zásahu a zvlášť výstupu ze soustavy, neboť bez tohoto by oba průběhy splynuly v jeden.



Obr. 3.11: Porovnání přenosů šumu u PID, PI*D a PSD regulátorů.



Obr. 3.12: Porovnání přenosů šumu u I-PD, I-P*D a S-PD regulátorů.

Z předcházejících dvou obrázků vidíme, že i když má PI*D regulátor relativně vysoké N, přesto má nejlepší vlastnosti co se týče přenosu šumů. Zároveň ale z obr. 3.11 vidíme, že jeho průběh regulace je nejvíce kmitavý.

Pokud porovnáme PSD a PID regulátor, zjistíme že PSD regulátor má lepší filtrační vlastnosti než PID regulátor s $N = 7$ a je porovnatelný s PID regulátorem s $N = 6$.

4. PŘEPÍNÁNÍ MEZI ALGORITMY REGULÁTORU JEDNOHO TYPU

4.1 ÚVOD

Naprogramujeme-li ve skriptu MATLABu regulátor a soustavu, provedeme jednoduchou simulaci, necháme soustavu vyregulovat na hodnotu 5V a zapíšeme si hodnoty jednotlivých složek regulátorů po ustálení, obdržíme následující tabulky. Regulátor má nastaveny parametry: $K = 2$, $T_I = 3$, $T_D = 0{,}5$, $N = 7$ a $T_T = 7$.

Tab. 4.1: Hodnoty jednotlivých složek PID regulátoru a jeho variant.

	Prop	Suma	Der	Ds
PID	0	5	0	0
PI-D	0	5	0	-6,64
I-PD	-10	15	0	-6,64

Tab. 4.2: Hodnoty jednotlivých složek PI*D a I-P*D regulátoru.

	Prop	Suma	Ds	Der1	Der2
PI*D	0	5	-5,78	0	5
I-P*D	-10	15	-5,78	0	5

Pro PSD a S-PD regulátor je zbytečné vytvářet tabulku, neboť algoritmus výpočtu je pro oba případy stejný, pouze jinak přistupujeme ke špičkám z prvních dvou kroků regulace.

Jak vidíme z průběhu na obr. 2.3 samotný náraz trvá po tak dlouhou dobu, než se jednotlivé složky regulátoru přenastaví z původních hodnot na nové. Pro co nejmenší náraz je potřeba tuto dobu minimalizovat, nejlépe ji zcela odstranit.

V následujícím textu budou uvedeny rovnice ve kterých figurují proměnné *Ds* a *Suma* někdy s indexem *k* a *k-1* a někdy ne. Pokud se tyto indexy v rovnici objevují, znamená to že pro výsledek výpočtu s indexem *k* potřebujeme znát předchozí

hodnotu této proměnné (index *k-1*). U některých výpočtů těchto proměnných předcházející stav znát nepotřebujeme a proto u něj žádný index neuvedeme.

4.2 PŘEPNUTÍ PID – PI-D

Rozdíl mezi těmito dvěma algoritmy spočívá v různé hodnotě proměnné *Alfa*. U PID je *Alfa* = 1 a u PI-D je *Alfa* = 0. Tato změna se projeví při výpočtu hodnoty proměnné Ds.

$$Ds_k = (Alfa \cdot w - y) + Ds_{k-1} \cdot \exp(-Tvz \cdot N / Td) \tag{4.1}$$

To nám potvrdí i výsledky z tab. 4.1. S hodnotou *Ds* se počítá v rovnici derivační složky

$$Der = N \cdot K \cdot ((Alfa \cdot w - y) - Ds + Ds \cdot \exp(-Tvz \cdot N / Td)) \tag{4.2}$$

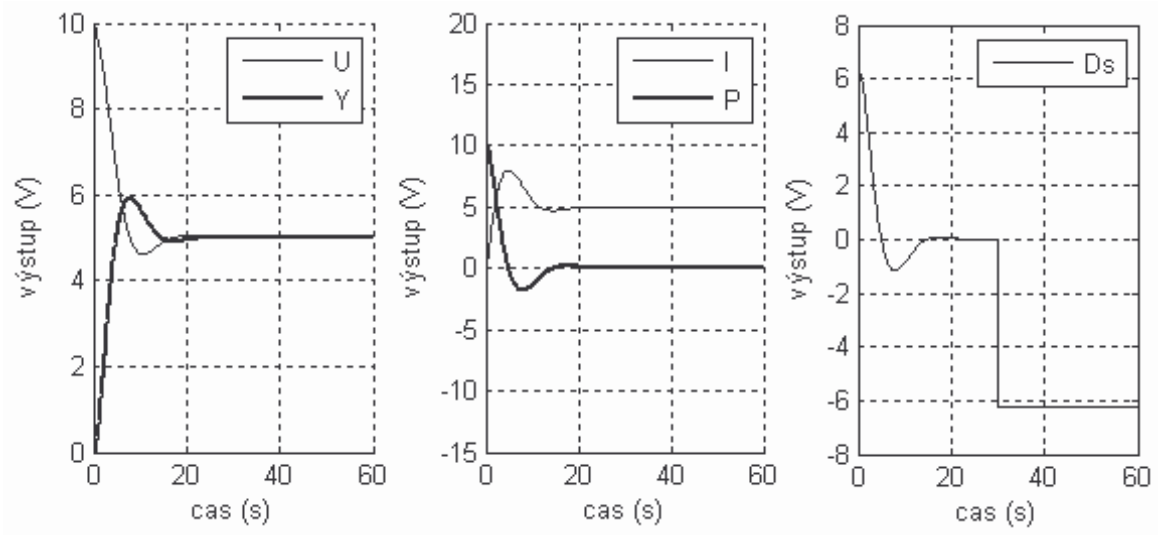
V ustáleném stavu, kdy se mohou přepínat varianty regulátoru, se má derivační složka rovnat nule. U PID varianty je toto snadno docíleno rovností $w = y$ a tím se celé *Ds* rovná nule. V případě PI-D varianty to ale bude odlišné, neboť se nám zde vůbec nevyskytuje žádaná hodnota *w*. *Ds* tedy bude vždy jisté záporné číslo. Aby se tedy derivační složka regulátoru rovnala nule, musí platit

$$-y - Ds + Ds \cdot \exp(-Tvz \cdot N / Td) = 0 \tag{4.3}$$

Po vyjádření *Ds* dostaneme

$$Ds = \frac{-y}{1 - \exp(-Tvz \cdot N / Td)} \tag{4.4}$$

Pokud pomocí tohoto vzorce změníme hodnotu *Ds* v okamžiku hned po zjištění požadavku na přepnutí z PID algoritmu na PI-D, dojde k beznárazovému přepnutí.



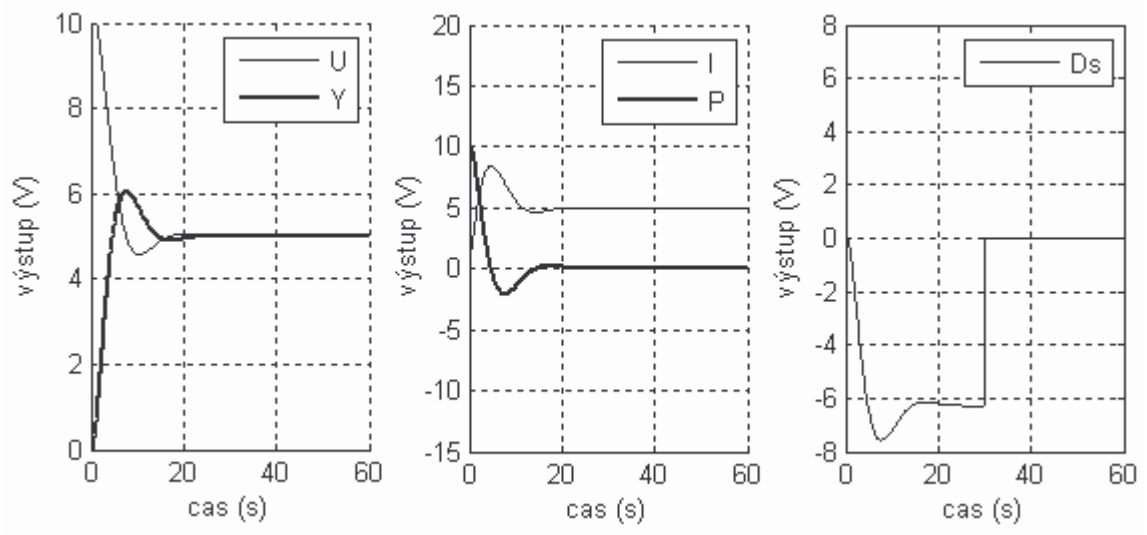
Obr. 4.1: Beznárazové přepnutí z PID na PI-D.

Pro opačné přepnutí, tedy z PI-D na PID je potřeba do *Ds* vložit rozdíl žádané hodnoty a aktuální hodnoty z výstupu soustavy.

$$Ds = w - y \tag{4.5}$$

Tyto dvě rovnice (4.4) a (4.5) se dají sloučit do jedné rovnice

$$Ds = Alfa \cdot (w - y) + (1 - Alfa) \cdot (-y) / (1 - \exp(-Tvz \cdot N / Td)) \tag{4.6}$$



Obr. 4.2: Beznárazové přepnutí z PI-D na PID.

4.3 PŘEPNUTÍ PID – I-PD

Oproti předchozímu přepnutí a změně hodnoty proměnné *Alfa* se v tomto případě mění i hodnota proměnné *Beta*. Tato změna se projeví ve výpočtu proporcionální složky regulátoru ve vzorci

$$Prop = K \cdot (Beta \cdot w - y) \tag{4.7}$$

I v této proměnné je v ustáleném stavu hodnota 0 a při změně hodnoty proměnné *Beta* z 1 na 0 se změní na

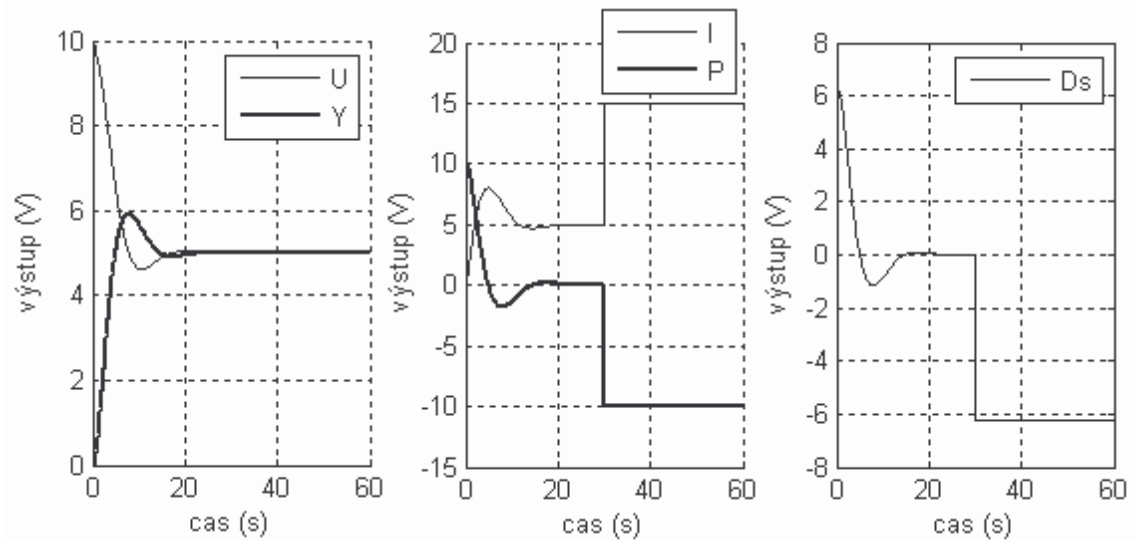
$$Prop = K \cdot (-y) \tag{4.8}$$

V našem případě nastavení konstant regulátoru to znamená číslo 2*(-5) = -10. Náraz by se v tomto případě projevil skokovou změnou akčního zásahu z hodnoty vypočítané vzorcem

$$U = Prop + Suma + Der \tag{4.9}$$

0 + 5 + 0 = 5 na -10 + 5 + 0 = -5. Je tedy zřejmé, že abychom dosáhli beznárazového přepnutí, musíme těchto 10 vhodně k regulátoru připočíst. Z tab. 4.1 vidíme, že se tato hodnota naváže na sumační složku regulátoru. Pro beznárazové přepnutí, tedy kromě změny popsané vzorcem (4.4), musíme do části programu reagující na požadavek na změnu algoritmu regulátoru přidat další přepočet

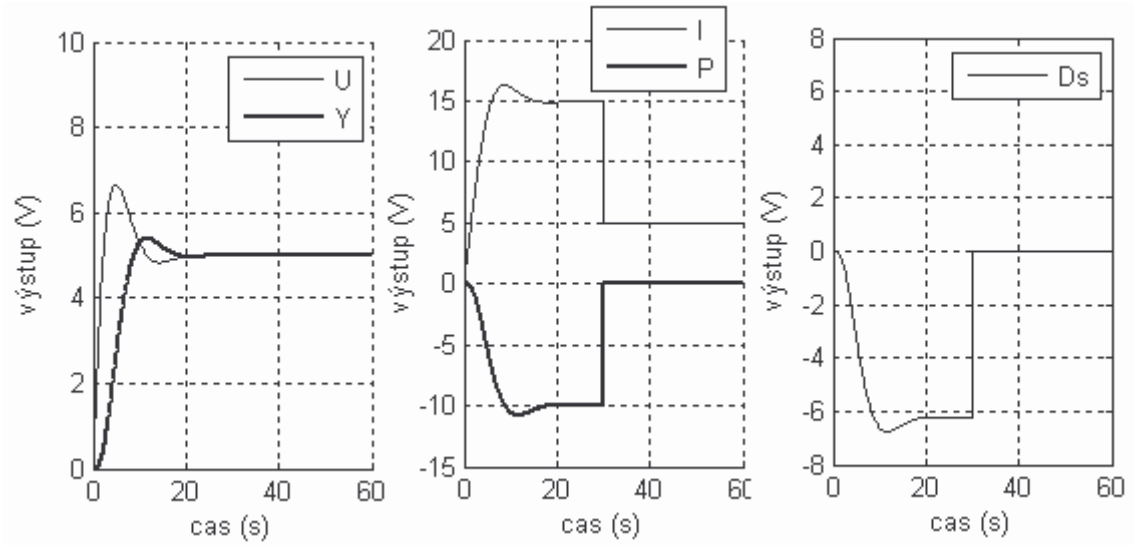
$$Suma_k = Suma_{k-1} + K \cdot y \tag{4.10}$$



Obr. 4.3: Beznárazové přepnutí z PID na I-PD.

Pro opačné přepnutí z I-PD na PID je opět potřeba hodnotu *Ds* zaměnit dle vzorce (4.5) a z proměnné *Suma* opět součin *K* * *y* odečíst pomocí rovnice

$$Suma_k = Suma_{k-1} - K \cdot y \tag{4.11}$$



Obr. 4.4: Beznárazové přepnutí z I-PD na PID.

Vzorce (4.10) a (4.11) lze spojit do jednoho za předpokladu, že budeme znát zároveň aktuální hodnotu proměnné *Beta* i její budoucí hodnotu po přepnutí na jiný

algoritmus. Tuto budoucí hodnotu budeme mít uloženou v proměnné *NovaBeta*. Společný vzorec by vypadal

$$Suma_k = Suma_{k-1} - (NovaBeta - Beta) \cdot K \cdot y \qquad (4.12)$$

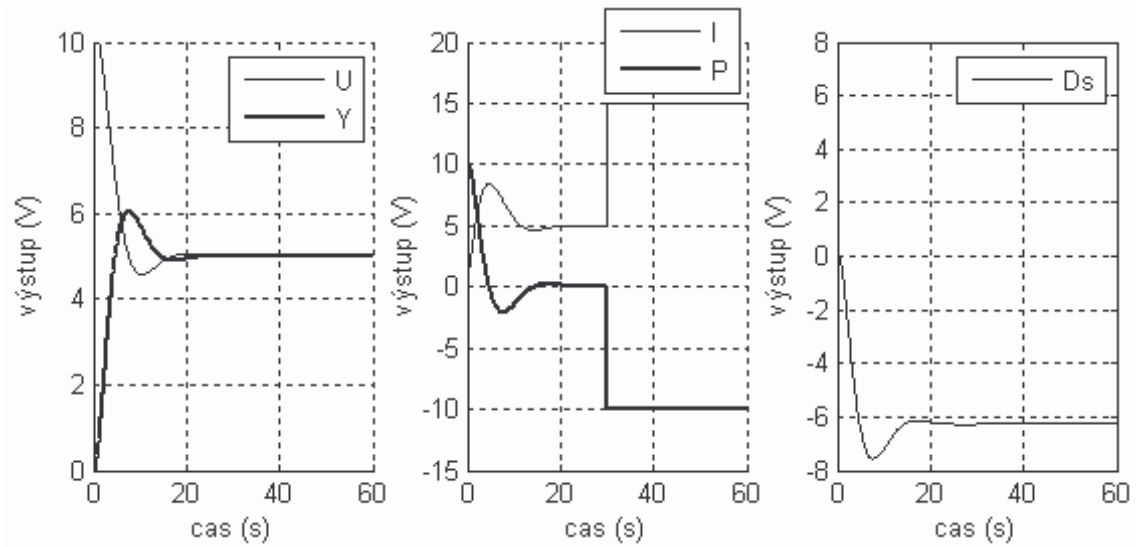
Tento vzorec by nabral na důležitosti, pokud bychom kromě regulátorů s pevně zvolenou betou (PI-D, I-PD) chtěli beznárazově přepínat i na regulátory, které jsou svými vlastnostmi mezi PI-D a I-PD regulátorem a mají libovolnou betu v rozmezí 0 ÷ 1, kterou zadáváme až přímo těsně před přepnutím na tento algoritmus.

4.4 PŘEPNUTÍ PI-D – I-PD

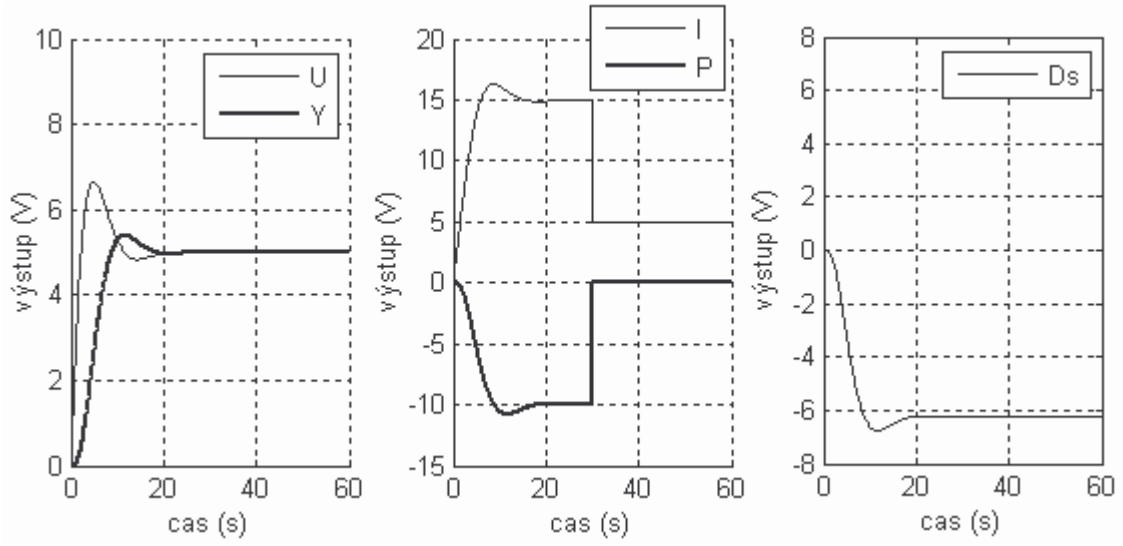
Problém přepnutí těchto dvou algoritmů je takřka totožný s předcházejícím typem přepnutí algoritmů pouze s tím rozdílem, že nemusíme řešit změnu derivační složky. Jediný problém tedy máme změnu hodnoty proporcionální složky regulátoru z nulové na zápornou.

Toto vyřešíme úplně stejně jako v předcházejícím případě buď zvlášť pomocí vzorců (4.10) a (4.11) nebo pomocí jednoho (4.12).

Průběhy změn algoritmů jsou zobrazeny na obr. 4.5 a 4.6



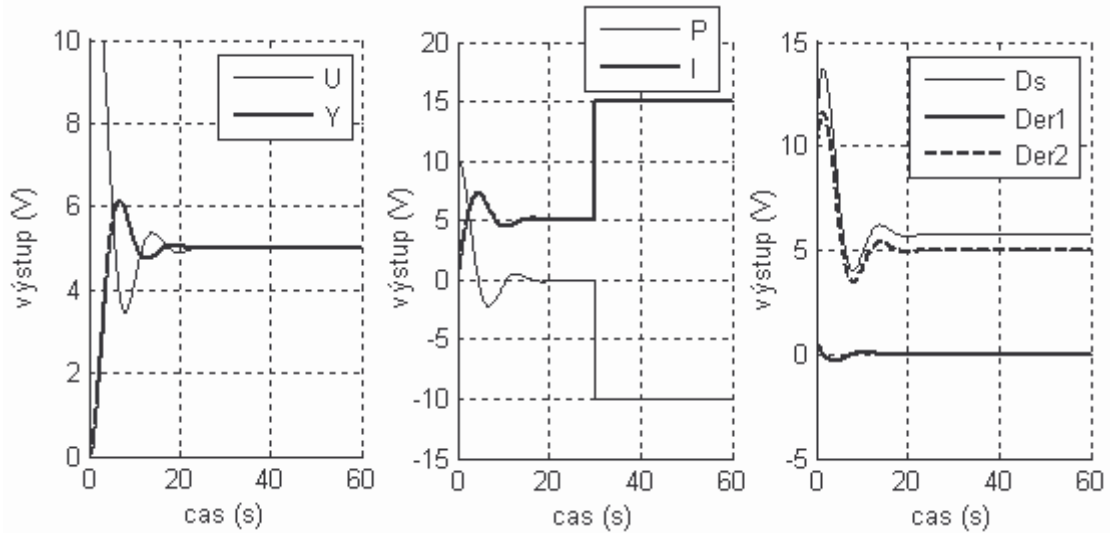
Obr. 4.5: Beznárazové přepnutí z PI-D na I-PD.



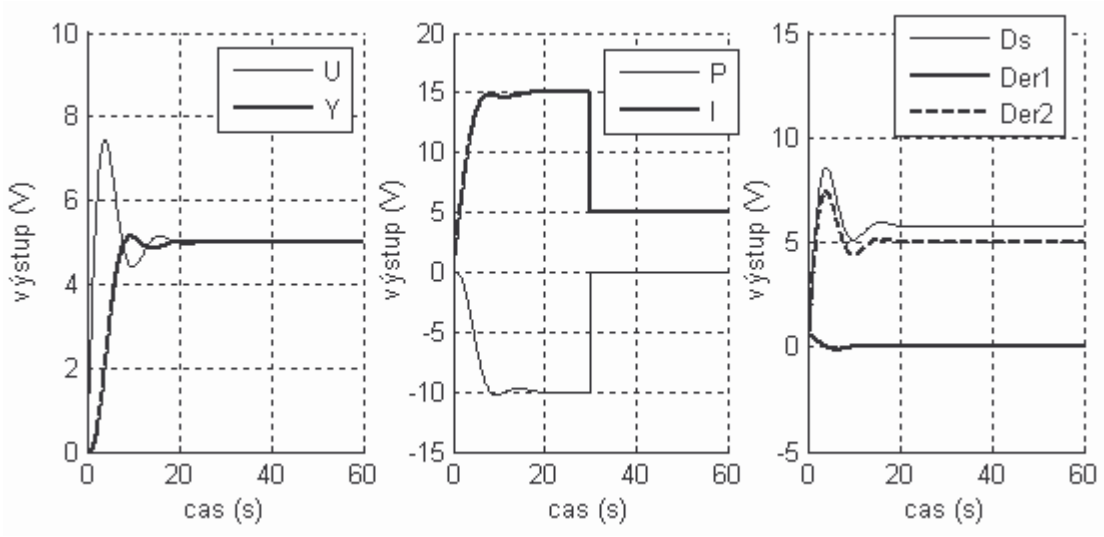
Obr. 4.6: Beznárazové přepnutí z I-PD na PI-D.

4.5 PŘEPNUTÍ PI*D – I-P*D

Porovnáním tabulky 4.1 s tabulkou 4.2 můžeme vyvodit závěr, že přepnutí provedeme stejným způsobem jako při přepnutí z PI-D na I-PD regulátor. Změnu hodnoty v sumační složce provedeme dle vztahu (4.12). Hodnoty v proměnných *Ds*, *Der1* a *Der2* není třeba měnit.



Obr. 4.7: Beznárazové přepnutí z PI*D na I-P*D.



Obr. 4.8: Beznárazové přepnutí z I-P*D na PI*D.

4.6 PŘEPNUTÍ PSD – S-PD

Jak už jsme psali v kapitole 3.3, tyto dva algoritmy vycházejí ze stejných rovnic. Rozdíl je pouze ve způsobu ošetření špiček z prvních dvou kroků regulace po změně žádané hodnoty. Pro toto přepnutí tedy není potřeba žádných rovnic, neboť hodnoty vnitřních proměnných jsou v ustáleném stavu stejné pro PSD i S-PD.

4.7 SHRNUTÍ

Jak vidíme, veškeré beznárazové přepínání mezi jednotlivými algoritmy regulátoru můžeme realizovat pouze díky dvou rovnic (4.12) a (4.6), pokud před těmito dvěmi rovnicemi budou nadefinovány proměnné *Alfa* a *NovaBeta* pro nový algoritmus. V některých případech si vystačíme pouze s jednou rovnicí a někdy nepotřebujeme žádnou.

5. PŘEPÍNÁNÍ MEZI AUTOMATICKÝM A MANUÁLNÍM REŽIMEM

5.1 PŘEPNUTÍ NA MANUÁLNÍ REŽIM

Musíme splnit pouze jediný požadavek, abychom mohli přepnout z automatického režimu na manuální. Tím je pokračování posledního akčního zásahu z automatického režimu.

Pokud tedy v obsluze požadavku na přepnutí režimů zajistíme, aby bylo toto splněno, nic jiného se již programem řešit nemusí. Proto si zavedeme proměnnou *ZAutNaMan*, kterou budeme používat pro indikaci přepnutí na manuální režim. Na tuto proměnnou si v manuálním režimu postavíme podmínku, která pokud je splněna, vložíme do proměnné reprezentující operátorovo manuální nastavení akčního zásahu poslední akční zásah daný automatickým režimem a zároveň zajistíme, aby se tato část kódu manuálního režimu již nespustila. V automatickém režimu musíme poté opět zajistit, aby byla při dalším přepnutí na manuální režim tato podmínka opět splněna. Námi navržený kód může vypadat například takto

```
{Manualni rezim}
IF ZAutNaMan=1 THEN
 MujAkcniZasah:=REAL_TO_INT(U*10);
 ZAutNaMan:=0;
END_IF
U:=MujAkcniZasah*0.1;
...
{Automaticky rezim}
ZAutNaMan:=1;
...
```

5.2 PŘEPNUTÍ NA AUTOMATICKÝ REŽIM

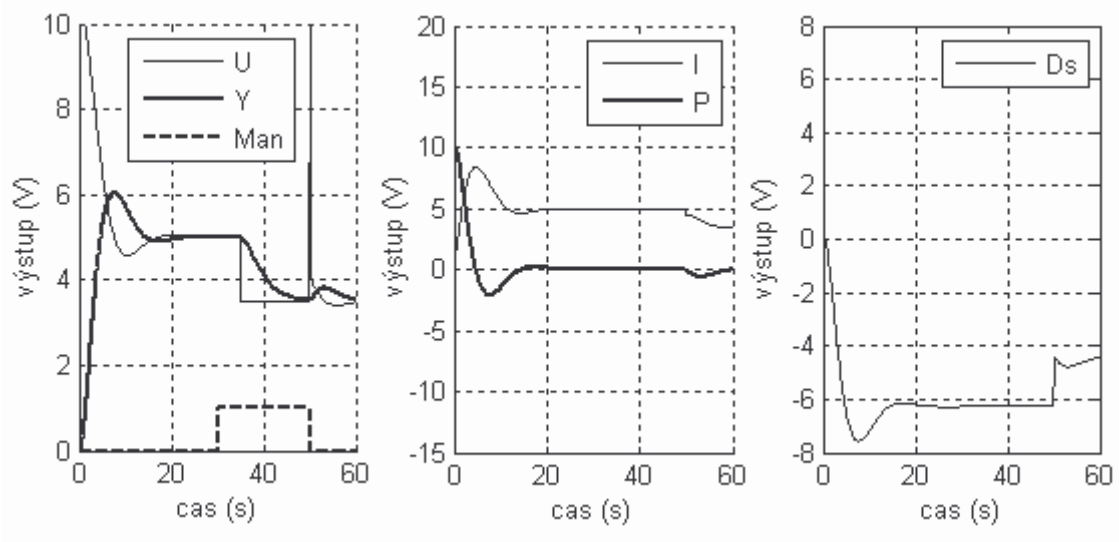
5.2.1 Popis přepnutí

Tento typ přepnutí se velice podobá předchozímu typu. Nyní nás namísto zjišťování posledního automatického akčního zásahu bude zajímat na jakou žádanou hodnotu bude po přepnutí na automatický režim algoritmus regulovat. Protože vycházíme z předpokladu, že k přepnutí mezi režimy dojde v ustáleném stavu ve

kterém platí $w = y$, použijeme právě výstupní hodnotu ze soustavy jako žádanou hodnotu pro automatický režim regulátoru.

Provedeme následující simulaci ve skriptu:

1. Automatickým režimem regulátoru nastavíme výstupní hodnotu ze soustavy na 5V
2. Po 30s přepneme na manuální režim
3. Po 5s snížíme velikost akčního zásahu a necháme ustálit výstupní hodnotu ze soustavy
4. Přepneme na automatický režim



Obr. 5.1: Nárazové přepnutí z manuálního na automatický PI-D režim.

Jak vidíme z obr. 5.1, pouze nastavení nové žádané hodnoty nestačí k beznárazovému přepnutí z manuálního na automatický režim.

Proto musíme zajistit, aby se při přepnutí z manuálního režimu do automatického vhodně změnily hodnoty složek regulátoru. Protože už ale z tab. 4.1 a 4.2 víme, že při různých algoritmech regulátoru mají jednotlivé složky odlišné hodnoty, musíme mít tolik výpočtových rovnic pro změnu hodnot složek kolik máme algoritmů regulátoru.

5.2.2 Určení výpočtových rovnic pro PID varianty algoritmů.

Pro PID algoritmus je určení Ds snadné a je stejné jako v rovnici (4.5): $Ds = w - y$.

Rovněž pro PI-D a I-PD se pro určení *Ds* používá již námi vypočtenou rovnici (4.4):

$$Ds = -y/(1-\exp(-Tvz \cdot N/Td))$$

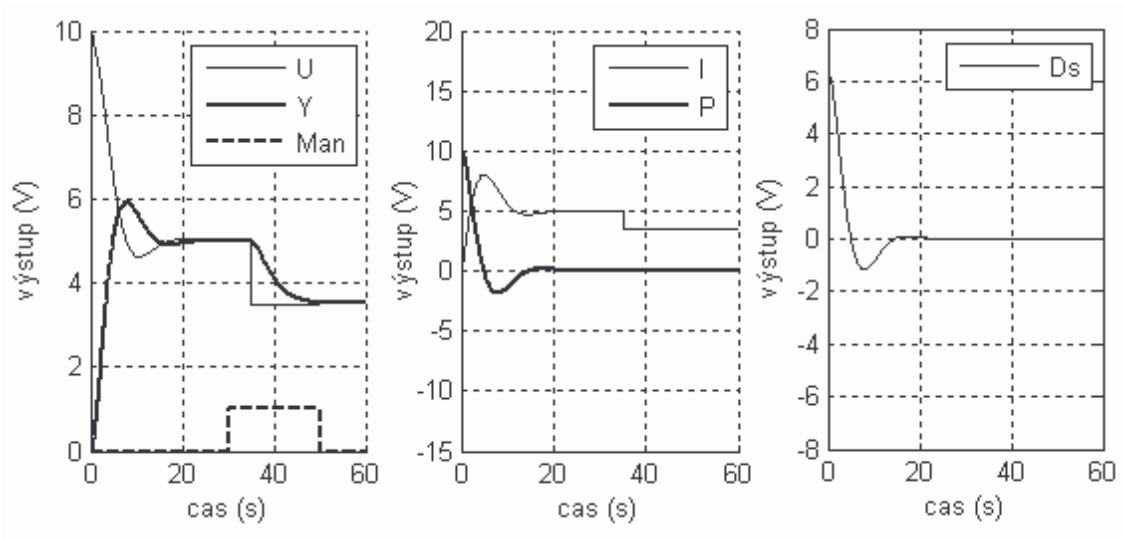
U PID a PI-D regulátorů platí v ustáleném stavu, že proporcionální a derivační složky jsou rovny nule a veškeré výstupní napětí je na integrátoru. Proto pro tyto dva algoritmy použijeme výpočtových rovnic ve znění

$$Suma = u \tag{5.1}$$

Poslední výpočtovou rovnici, kterou je potřeba zjistit, je výpočet pro proměnnou Suma u I-PD algoritmu. Zde bude platit vztah podobný rovnici (4.10). I zde je potřeba do proměnné *Suma* vložit kromě akčního zásahu *u* i součin *K * y* , aby se odstranilo působení záporné hodnoty *K * (-y)* z proporcionální složky regulátoru.

Konečný tvar výpočtových rovnic tedy bude vypadat (strukturovaný text)

```
{ PID Beta=1, Alfa=1}
IF RezimR=1 THEN Suma:=U; Ds:=W-Y; END_IF
{PI-D Beta=1, Alfa=0}
IF RezimR=2 THEN Suma:=U; Ds:=-Y/(1-EXP(-Tvz*N/Td)); END_IF
{I-PD Beta=0, Alfa=0}
IF RezimR=3 THEN Suma:=U+K*Y; Ds:=-Y/(1-EXP(-Tvz*N/Td)); END_IF
```



Obr. 5.2: Beznárazové přepnutí z manuálního na automatický PID režim.

Poslední tři zmíněné programové podmínkové výrazy můžeme opět velice jednoduše zjednodušit pouze do dvou přiřazovacích rovnic

```
Suma:=U+(1-Beta)*K*Y;
Ds:=Alfa*(W-Y)+(1-Alfa)*(-Y)/(1-EXP(-Tvz*N/Td));
```

Vytvořili jsme tedy nový univerzální vztah:

$$Suma = u + (1 - Beta) \cdot K \cdot y \tag{5.2}$$

5.2.3 Určení výpočtových rovnic pro PI*D varianty algoritmů.

Porovnáním sumačních složek v tab. 4.1 a 4.2 pro PI-D a PI*D algoritmy vypozorujeme, že jeho hodnoty v ustáleném stavu jsou obou případech stejné, neboť v algoritmu regulátoru se sumační složka počítá stejně pro oba případy a z toho vyvodíme, že algoritmus pro beznárazové přepnutí na automatický režim pro tuto složku bude stejný. Tedy

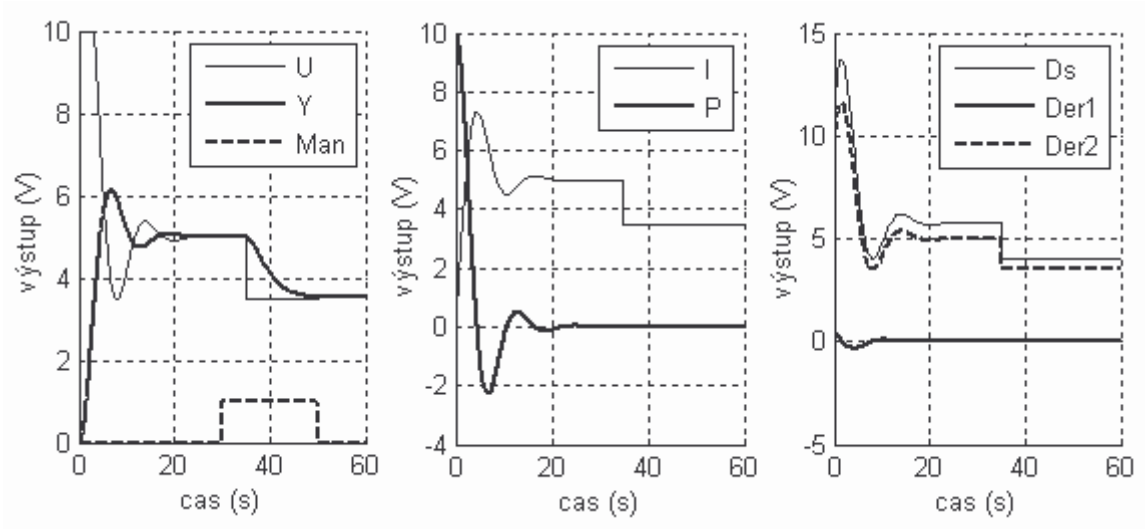
```
Suma:=U+(1-Beta)*K*Y;
```

Avšak u proměnných *Ds* jsou jiné hodnoty. Důvod je jasný: u PI-D algoritmu vstupuje do derivační složky záporně vzatá hodnota výstupu ze soustavy *–y* a u PI*D algoritmu výstupní hodnota z PI části. Rovnici (4.4) tedy upravíme na

$$Ds = Upi / (1 - \exp(-Tvz \cdot 10 / Td)) \tag{5.3}$$

Třetí věc, kterou je potřeba zaručit je změna parametrů pro setrvačný článek v derivační části závorky. Celý algoritmus tedy bude vypadat

```
Upi:=U;
R_U1:=U;
R_E1:=U;
Suma:=U+(1-Beta)*K*Y;
Ds:=Upi/(1-EXP(-Tvz*10/Td));
```

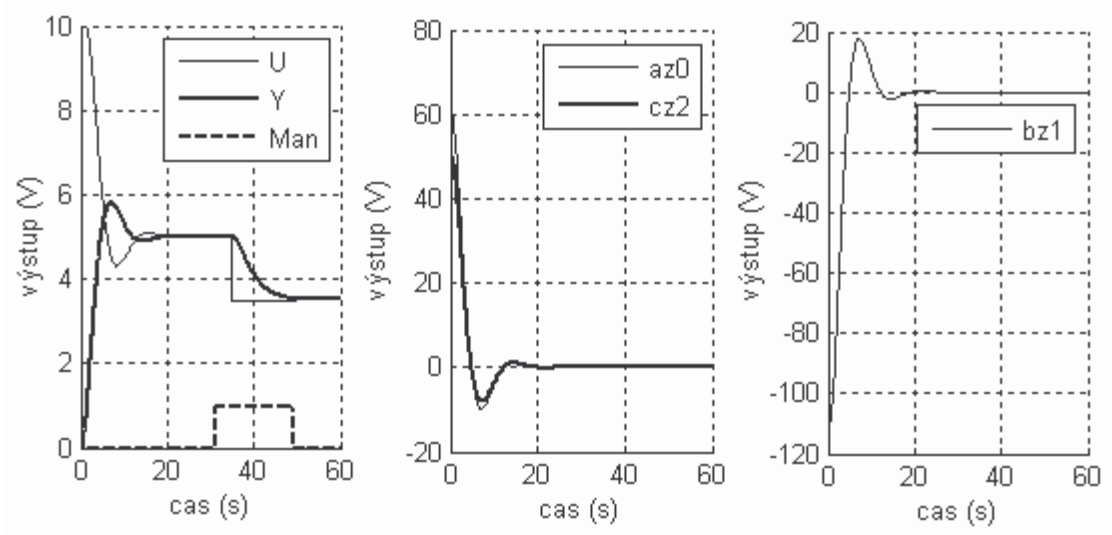


Obr. 5.3: Beznárazové přepnutí z manuálního na automatický PI*D režim.

5.2.4 Určení výpočtových rovnic pro PSD varianty algoritmů.

U PSD regulátoru je přepnutí nejjednodušší. Nemusíme nic počítat, protože víme, že hodnoty proměnných *R_E0*, *R_E1* a *R_E2* jsou v ustáleném stavu rovny nule, neboť regulační odchylka *e* je rovna nule a hodnota proměnné *R_U1* sleduje hodnotu akčního zásahu.

```
R_U1=U;
R_E0=0;
R_E1=0;
R_E2=0;
```



Obr. 5.4: Beznárazové přepnutí z manuálního na automatický PSD režim.

6. PŘEPÍNÁNÍ MEZI ROZDÍLNÝMI TYPY REGULÁTORŮ

V minulé kapitole jsme objasnili beznárazové přepínání z libovolného algoritmu regulátoru do manuálního režimu a poté následně zase do libovolného algoritmu. Můžeme tedy konstatovat, že pokud vynecháme přepnutí na manuální režim, budeme moct přepnout z libovolného algoritmu na jakýkoliv jiný. Zjištěné algoritmy z minulé kapitoly tedy použijeme pro veškeré do této doby zmíněné přepnutí.

Zavedeme tedy programovou podmínku ve znění

```
IF ((ZManNaAut=1) OR ((Cas > 50) AND (NovyRez>0))) THEN
...
END_IF
```

, v jejímž těle budou právě výše zmíněné přepínací algoritmy. Ty se provedou pokud buď přepínáme z manuálního režimu `(ZManNaAut=1)`, nebo chceme přepnout na jiný algoritmus `(NovyRez>0)` a zároveň je ustálený výstup ze soustavy `(Cas > 50)`.

Na tomto místě upozorníme, že na vztah (4.12)

$$Suma_k = Suma_{k-1} - (NovaBeta - Beta) \cdot K \cdot y$$

, který jsme odvodili v kapitole 4.3 můžeme zcela zapomenout, neboť jej beze zbytku nahradil vztah (5.2)

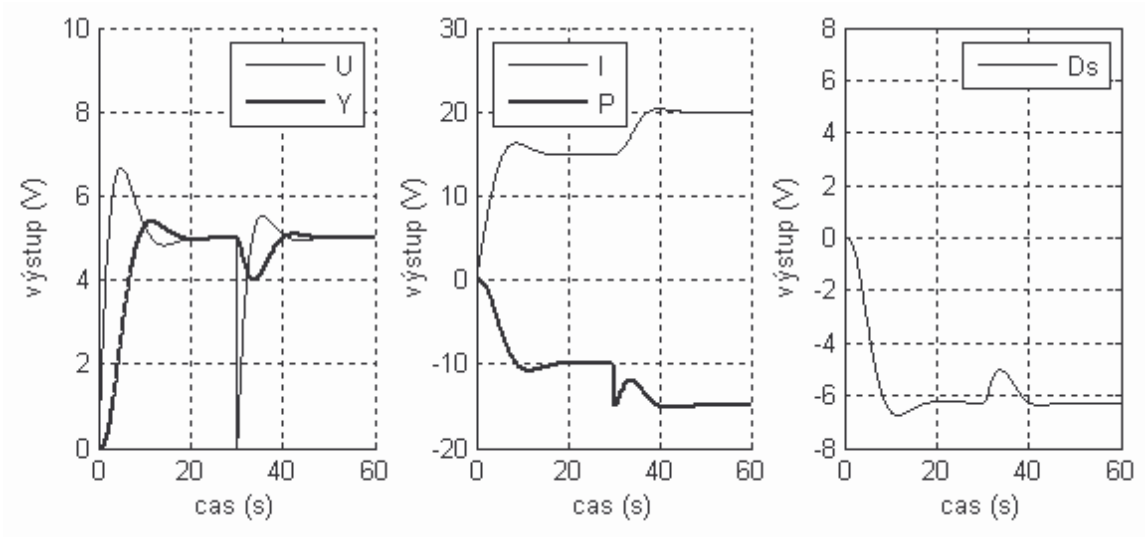
$$Suma = u + (1 - Beta) \cdot K \cdot y$$

Tímto jsme se zbavili nadbytečné proměnné *NovaBeta*.

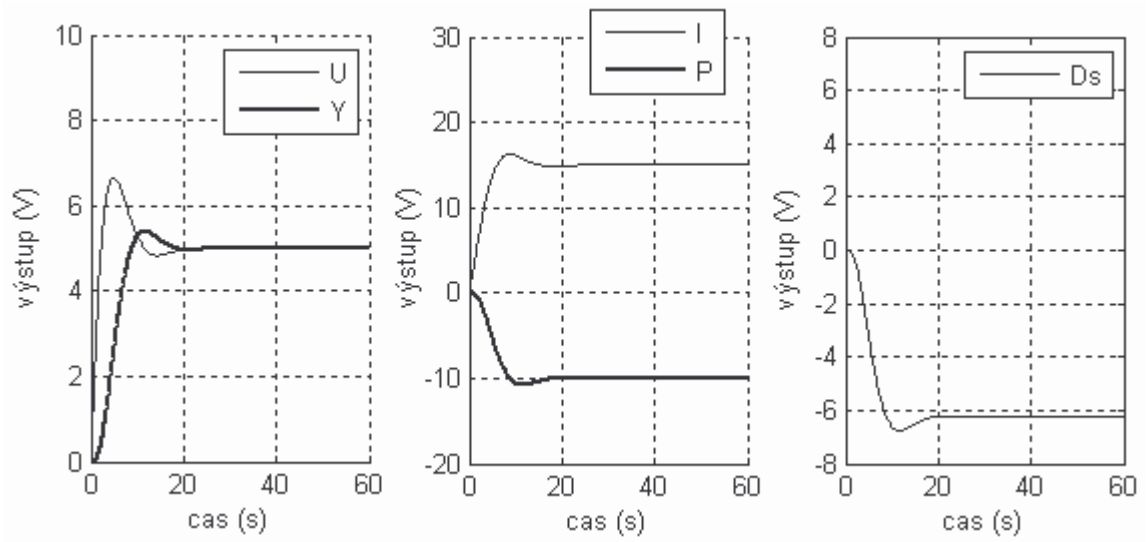
7. ZMĚNA HODNOTY VNITŘNÍ KONSTANTY REGULÁTORU (K, T_D, N) NA JINOU.

7.1 ÚVOD

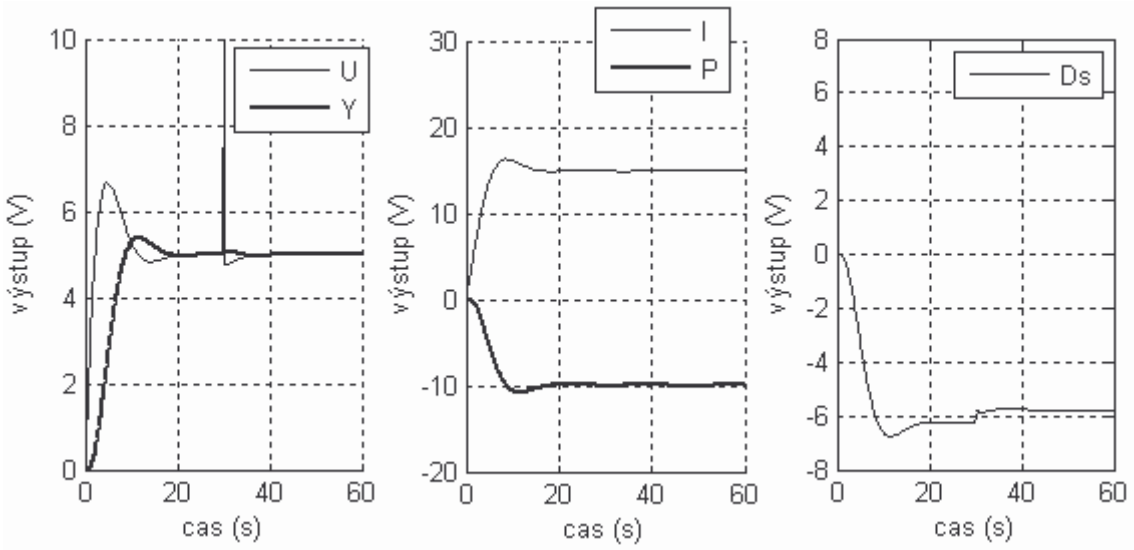
Nejprve si na grafech odezvy na skokovou změnu žádané hodnoty a následném ustálení ukážeme na I-PD regulátoru vlivy změn hodnot jednotlivých parametrů.



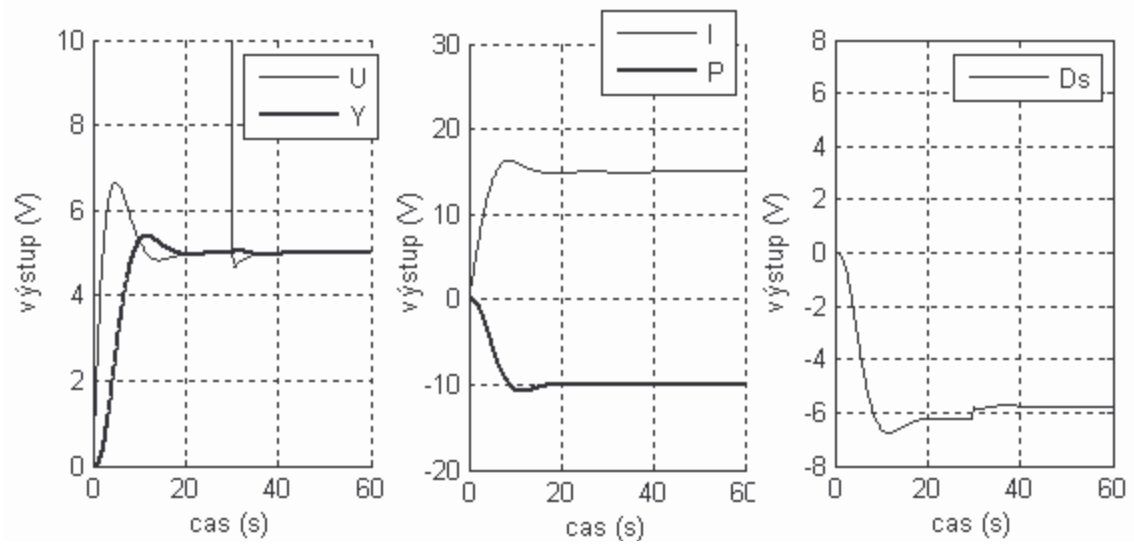
Obr. 7.1: Vliv skokové změny parametru K v regulátoru.



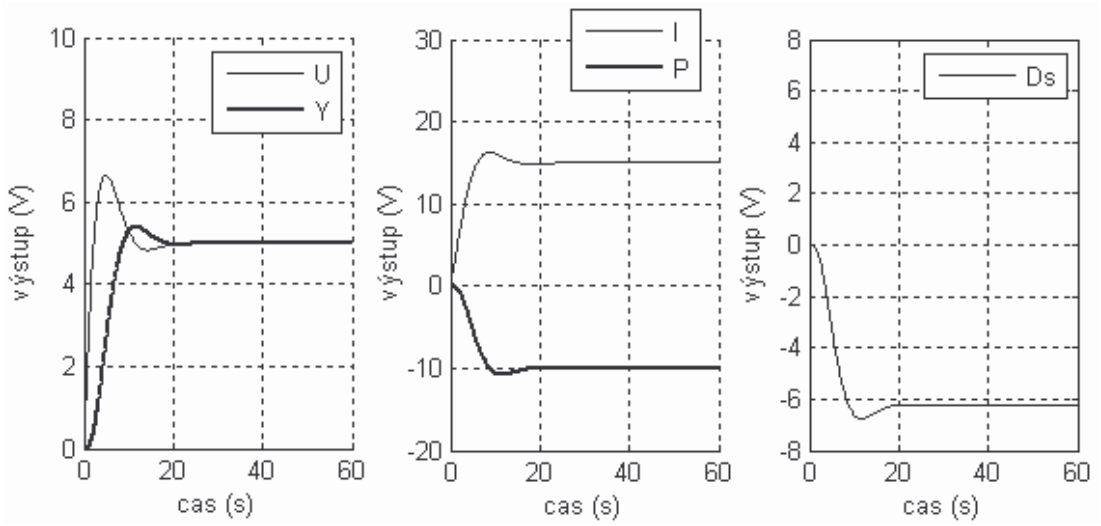
Obr. 7.2: Vliv skokové změny parametru Ti v regulátoru.



Obr. 7.3: Vliv skokové změny parametru Td v regulátoru.



Obr. 7.4: Vliv skokové změny parametru N v regulátoru.



Obr. 7.5: Vliv skokové změny parametru Tt v regulátoru.

Z předchozích obrázků jde vidět, že změny proměnných *Ti* a *Tt* nemají žádný vliv na regulátor ani soustavu při ustáleném stavu. Skoková změna *N* a *Td* se projeví nárazem u algoritmů s hodnotou proměnné *Alfa* = 0, tedy PI-D a I-PD. U PI*D a PSD alternativ regulátorů se proměnná *N* nevyskytuje, náraz tedy vzniknout nemůže. U PI*D regulátorů však vznikne náraz při změně proměnné *Td*. Pokud v průběhu regulace změníme hodnotu parametru K a budeme mít nastavený algoritmus PID, PI*D nebo PSD, náraz se neprojeví. Pokud ale budeme mít nastavený regulátor, kde proměnná *Beta* je různá od 1, což jsou v našem případě algoritmy I-PD a I-P*D, náraz se projeví.

7.2 ODSTRANĚNÍ NÁRAZU

Nejjednodušší odstranění nárazů způsobenými změnou parametrů N a Td spočívá v okamžitém přepočítání proměnné Ds na novou s použitím nových hodnot N nebo Td podle vzorce (4.4) pro PI-D a I-PD:

$$Ds = -y/(1-\exp(-Tvz \cdot N/Td))$$

a podle vzorce (5.3) pro PI*D a I-P*D:

$$Ds = Upi/(1-\exp(-Tvz \cdot 10/Td))$$

Jak vidíme z obr. 7.3 a 7.4, změny *N* a *Td* nemají vliv ani na sumační ani na proporcionální složku regulátoru.

Náraz vytvořený změnou parametru *K* je způsoben pomalou změnou sumační složky, jenž musí vyrušit působení záporné hodnoty z proporcionální složky, která se po změně parametru *K* skokově změnila. K eliminaci tohoto budeme postupovat stejně jako při řešení problému přepnutí z manuálního na automatický režim. Do proměnné *Suma* vložíme stávající akční zásah *u* a součin *K* * *y*. Dostaneme tak již jednou použitý vzorec

$$Suma = u + K \cdot y \tag{7.1}$$

7.3 SHRNUTÍ

Rovněž naposledy uvedený vztah lze upravit do takové podoby, aby stačil jeden vztah pro všechny typy algoritmů regulátoru. Proto rozšíříme součin *K* * *y* ještě o již několikrát použitý rozdíl (1 – *Beta*) a tím obdržíme vztah použitý i v přepnutí z manuálního do automatického režimu po zjednodušení výpočtových rovnic (5.2)

$$Suma = u + (1 - Beta) \cdot K \cdot y$$

Pro úplnou spolupráci se všemi algoritmy regulátoru také použijeme pro výpočet proměnné *Ds* místo vzorce (4.4) vztah (4.6):

$$Ds = Alfa \cdot (w - y) + (1 - Alfa) \cdot (-y) / (1 - \exp(-Tvz \cdot N / Td))$$

Jak vidíme, tyto rovnice jsou shodné s rovnicemi, které jsme označili za výsledné u přepnutí z manuálního do automatického režimu. Proto pokud zajistíme, aby se mohly měnit parametry regulátoru pouze v manuálním režimu a v automatickém bude toto znemožněno, potom není potřeba vůbec řešit tuto alternativu nárazového přepnutí, neboť se vyřeší již odstraněním nárazu přepnutí z manuálního na automatický režim.

8. SIMULACE VE SKRIPTU MATLABU

Nejdříve je potřeba si zvolit délku simulace. Pokud si zvolíme 100s znamená to, že dohromady spočítáme 1000 iteračních kroků. To nám umožní použit naši soustavu odvozenou v kapitole 2.3 ve které se počítá se vzorkovací periodou T_{VZ} = 0,1s a tedy jeden iterační krok se bude rovnat 0,1s. Potom budou výsledné průběhy této simulace porovnatelné s real-time simulací jenž následně vytvoříme v B&R virtuálním řídicím systému.

Na začátku skriptu je potřeba vymazat všechny použité proměnné, aby nebyla nová simulace ovlivněna výsledky z minulého spuštění. Následně je potřeba nastavit parametry regulátoru pro simulaci.

Nyní vytvoříme smyčku o daném počtu iteračních kroků. Do ní vložíme regulátor, dynamické omezení sumační složky a akčního zásahu a soustavu. Zároveň průběžně ukládáme hodnoty pro grafické zobrazení simulace.

Do skriptu naprogramujeme všechny čtyři typy událostí, u kterých by běžně mohlo dojít k nárazu a hned za nimi použijeme rovnice, které jsme si v tomto textu odvodili, jenž tyto možné nárazy eliminují. Námi odvozené a použité výpočtové rovnice jsou vypsány v tab. 8.1.

Tab. 8.1: Výpočtové rovnice:

Výpočet *Sumy* pro všechny případy:
$Suma = u + (1 - Beta) \cdot K \cdot y$
Výpočet *Ds* pro PID, PI-D a I-PD algoritmy:
$Ds = Alfa \cdot (w - y) + (1 - Alfa) \cdot (-y) / (1 - \exp(-Tvz \cdot N / Td))$
Výpočet *Ds* pro PI*D a I-P*D algoritmy:
$Ds = Upi / (1 - \exp(-Tvz \cdot 10 / Td))$

Přepnutí na jiný typ algoritmu regulátoru jednoho typu je realizováno pomocí proměnných *Beta* a *Alfa*. Po zjištění požadavku na přepnutí regulátoru jsou složky regulátoru upraveny pomocí zjištěných rovnic.

```
if i==400
    Beta=1;
    Alfa=1;
    Suma=U+(1-Beta)*K*Y;
    Ds=Alfa*(W-Y)+(1-Alfa)*(-Y)/(1-exp(-Tvz*N/Td));
end
```

Přepnutí na manuální režim je prezentováno vypnutím algoritmu regulátoru, kdy poslední akční zásah zůstane nezměněn. Následně manuálně změníme velikost akčního zásahu a soustavu opět necháme ustálit a poté přepneme zpět na automatický režim. Úpravy velikosti složek regulátoru a žádané hodnoty pro automatický režim se provádějí dynamicky v části programu pro manuální režim.

```
if i==450  AutMan=1;  end      % Prepnuti na manualni rezim.
if i==500  U=U-1;     end      % Zmena akcniho zasahu.
if i==700  AutMan=0;  end      % Prepnuti na automaticky rezim.
```

Změna parametrů regulátoru je provedena změnou všech konstant a následným provedením výpočtových rovnic. Původní parametry byly $K = 2$, $Ti = 3$, $Td = 0{,}5$, $N = 7$ a $Tt = 7$.

```
if i==800
    K=1;
    Ti=5;
    Td=1.5;
    N=10;
    Tt=10;
    Suma=U+(1-Beta)*K*Y;
    Ds=Alfa*(W-Y)+(1-Alfa)*(-Y)/(1-exp(-Tvz*N/Td));
end
```

Poslední ukázka bude přepnutí z PID na I-P*D algoritmus.

```
if i==900
    Rezim=1;
    Upi=U;
    R_U1=U;
    R_E1=U;
    Suma=U+(1-Beta)*K*Y;
    Ds=Upi/(1-exp(-Tvz*10/Td));
end
```

Důležité části m-file bez vynulování proměnných a zobrazení grafů jsou ukázány v příloze 1.

9. SIMULACE POMOCÍ VIRTUÁLNÍHO ŘS B&R

Toto není návod jak programovat v programu Automation Studiu v.3 firmy B&R. Je vyžadována určitá znalost tohoto programu. Je potřeba, aby uživatel uměl definovat proměnné, vytvořit jednoduchý program, spustit virtuální ŘS AR000 a umět s ním komunikovat. Rovněž zde není popsán postup tvorby vizualizace. Pouze je v příloze 2 uveden popis ovládání obrazovek.

V úvodní *Init* části programu je potřeba stejně jako ve skriptu MATLABu vynulovat všechny složky regulátoru a nastavit parametry algoritmu regulátoru, který se spustí jako první.

Základem cyklického programu je rozlišení, zda se nachází regulátor v automatickém či manuálním režimu. Nezávisle na zvoleném režimu regulátoru probíhá výpočet výstupní hodnoty ze soustavy.

V manuálním režimu se nastavuje velikost akčního zásahu v rozmezí od 0 do 100%. Výstup ze soustavy je v rozmezí 0 až 10V, proto stačí naši velikost akčního zásahu v % vydělit 10.

Na začátku části kódu manuálního režimu je potřeba provést opatření pro beznárazové přepnutí z automatického na manuální režim popsaný v kapitole 5.1. Poté nastavíme velikost akčního zásahu a převedeme současný výstup ze soustavy na žádanou hodnotu v automatickém režimu.

```
U:=MujAkcniZasah*0.1;
ZadanaAuto:=Y;
```

V automatickém režimu je nejdříve zajistit, aby v tomto režimu nebylo možné změnit parametry regulátoru. Toto zařídíme pomocí proměnné *ZManNaAut*, jenž indikuje přepnutí a pomocí zálohovací proměnné *Konst*[].

```
IF ZManNaAut=1 THEN
      ZManNaAut:=0;
      Konst[0]:=K;
      Konst[1]:=Ti;
      Konst[2]:=Td;
      Konst[3]:=N;
      Konst[4]:=Tt;
ELSE
      K:=Konst[0];
      Ti:=Konst[1];
      Td:=Konst[2];
      N:=Konst[3];
```

```
        Tt:=Konst[4];
 END_IF
```

Na rozdíl od skriptu MATLABu, kdy jsme se po manuálním režimu vrátili zpět do původního automatického algoritmu, v tomto programu musí být zajištěno, aby přepnutí z manuálního režimu do jakéhokoliv automatického algoritmu proběhlo beznárazově. Proto doplníme hned za uložení konstant regulátoru jednotlivé kombinace proměnných *Alfa*, *Beta* a *TypPSD* pro všech sedm algoritmů regulátorů.

```
IF NovyRez=1 THEN (* PID *)
    Beta:=1;
    Alfa:=1;
END_IF
IF NovyRez=2 THEN (* PI-D *)
    Beta:=1;
    Alfa:=0;
END_IF
...
IF NovyRez=7 THEN (* S-PD *)
    TypPSD:=2;
END_IF
```

Následně doplníme kódy zajišťující přepočet proměnných dle kapitoly 5.2.

```
IF TypRegulatoru = 1 THEN
    Suma:=U+(1-Beta)*K*Y;
    Ds:=Alfa*(W-Y)+(1-Alfa)*(-Y)/(1-EXP(-N*Tvz/Td));
END_IF
...
IF TypRegulatoru = 3 THEN
    R_U1:=U;
    R_E0:=0;
    R_E1:=0;
    R_E2:=0;
END_IF
```

Další součást programu je hlídání ustálenosti výstupu ze soustavy. Výstup označíme za ustálený, pokud je odchylka *y - w* menší než námi zvolená mez po dobu 5s. Až po uplynutí této doby může dojít ke změně algoritmů regulátoru. Pokud se vytvoří požadavek na změnu algoritmu regulátoru reprezentovanou změnou hodnoty proměnné *NovyRez* z nuly na jinou, vytvoří se podmínky pro beznárazové přepnutí a provede se stejný kód jako při přepnutí z manuálního na automatický režim.

Nakonec doplníme automatický režim o regulátory samotné a dynamické omezení sumační složky a akčního zásahu, který jsme již předtím rozšířili o část přímo pracující s PSD regulátorem..

10. REGULACE REÁLNÉ SOUSTAVY

Program nahraný do ŘS bude takřka identický programu využitého v předcházející kapitole. Virtuální řídicí systém AR000 nahradíme Power Panelem typu 4PP480.1043-75, nakonfigurujeme X20 rozšiřující vstupně/výstupní karty a na odpovídající piny navážeme proměnné *Flow* - snímač průtoku a *Fan* - výstup na ventilátor.

Neboť rozsahu napětí 0 až 10V odpovídá rozsah hodnot 0 - 32768 a v regulátoru nastavujeme žádanou hodnotu ve voltech, musíme výstupní hodnotu ze soustavy dělit číslem 3276 a akční zásah spočtený regulátorem právě naopak násobit číslem 3276 a až toto číslo vložit do D/A převodníku výstupní X20 karty.

Zároveň do programu vložíme algoritmus virtuální soustavy a možnost na ní přepnout. To umožní názornou ukázku funkčnosti algoritmu v řídicím systému za předpokladu, že nebude možné připojit soustavu. Výběr mezi reálnou soustavou a virtuální řídí proměnná *SimReal*. Pro volbu mezi těmito dvěma soustavami musí být vložena část kódu

```
IF SimReal = 0 THEN
    Flow:=REAL_TO_INT(Y*3276);
END_IF
Y:=Flow;
Y:=Y/3276;
```

na začátku cyklického programu a část kódu

```
IF SimReal = 0 THEN
    Y:=0.0008105*U1+0.0007883*U2+1.9184012*Y1-0.92*Y2;
    U2:=U1;
    U1:=U;
    Y2:=Y1;
    Y1:=Y;
ELSE
    Fan:=REAL_TO_INT(U*3276);
END_IF
```

na konci.

Celé znění cyklického programu ve strukturovaném textu je v příloze 3.

11. ZÁVĚR

Cílem práce bylo nalézt vhodné rovnice, jenž by umožnily beznárazové přepínání mezi všemi popsanými algoritmy regulátoru, mezi automatickým a manuálním režimem a rovněž aby změna parametrů regulátoru byla provedena beznárazově. K těmto rovnicím jsme se dobrali pomocí postupného studia změn velikostí složek regulátoru, jenž jsme simulovali ve skriptu MATLABu. V průběhu vývoje skriptu se algoritmy beznárazového přepínání vylepšovaly a zjednodušovaly až do finální podoby tří výpočtových rovnic.

Kromě určení výpočtových rovnic bylo rovněž nutné určit správné okamžiky pro přepočítání složek regulátorů a rovněž zvolit správné podmínky, aby nebyly části kódu nadbytečné nebo zdvojené.

Bylo zjištěno, že při použití PSD algoritmu je potřeba kromě naprogramování vlastního regulátoru i vhodně programově ošetřit velké změny akčního zásahu v prvních dvou krocích algoritmu těsně po změně žádané hodnoty.

Regulátor v Power Panelu v takové podobě jako je dnes je pouze ukázkovou úlohou a jistě by díky námi navržené vizualizace neměl praktického použití v průmyslu. Pro použití ve výuce jako demonstrační prostředek by však použit být mohl.

Algoritmy zabezpečující beznárazové přepínání jsou však zcela funkční a jejich praktickému použití by nemělo nic bránit.

LITERATURA

[1] PIVOŇKA P.: *Číslicová řídicí technika*. VUT, Brno, 2003, E-text AMT101

[2] VELEBA V.: *Číslicová řídicí technika – Počítačová cvičení*. VUT, Brno, 2005, E-text

SEZNAM ZKRATEK

Zkratka/Symbol	Jednotka	Popis
ŘS		Řídicí systém
w	V	Žádaná hodnota
y	V	Okamžitá hodnota výstupu ze soustavy
e	V	Regulační odchylka
u	V	Velikost akčního zásahu
T_{VZ}	s	Vzorkovací perioda

SEZNAM PŘÍLOH

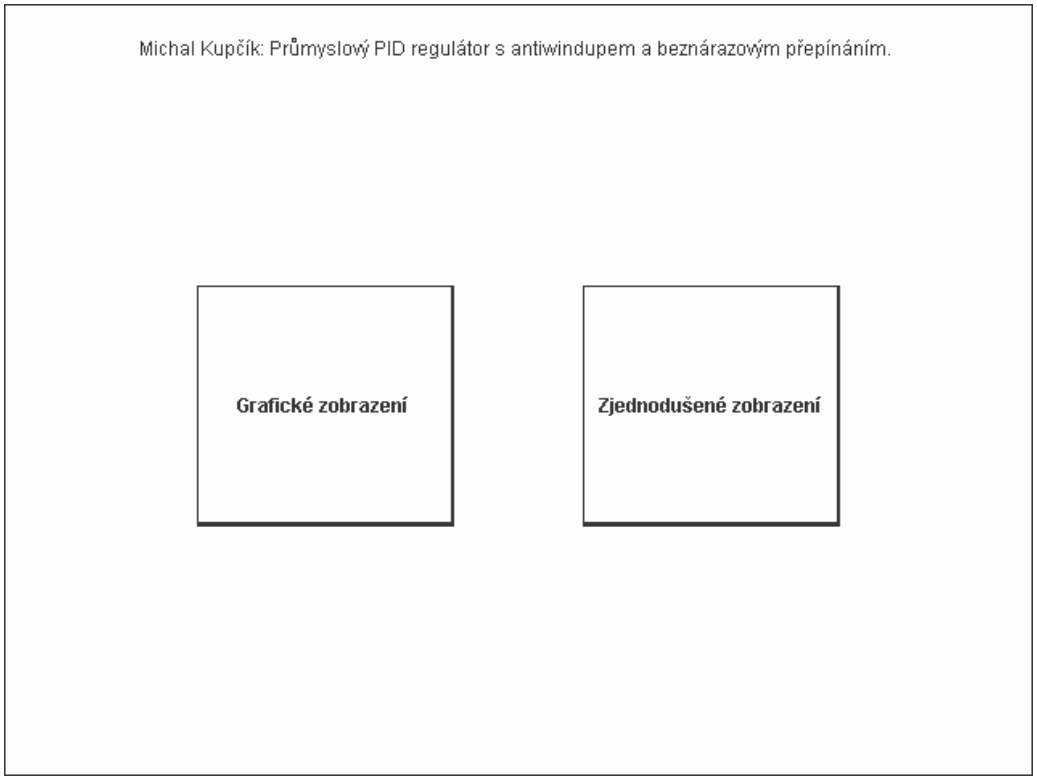
Příloha 1. Hlavní části simulačního skriptu MATLABu

```
for i=1:1:t

    if i==400
        Beta=0;
        Alfa=0;
        Suma=U+(1-Beta)*K*Y;
        Ds=Alfa*(W-Y)+(1-Alfa)*(-Y)/(1-exp(-Tvz*N/Td));
    end

    if i==450
        AutMan=1;  end      % Prepnuti na manualni rezim.
    if i==500
        U=U-1;     end      % Zmena akcniho zasahu.
    if i==700
        AutMan=0;  end      % Prepnuti na automaticky rezim.

    if i==800
        K=1;
        Ti=5;
        Td=1.5;
        N=10;
        Tt=10;
        Suma=U+(1-Beta)*K*Y;
        Ds=Alfa*(W-Y)+(1-Alfa)*(-Y)/(1-exp(-Tvz*N/Td));
    end

    if i==900
        Rezim=1;
        Upi=U;
        R_U1=U;
        R_E1=U;
        Suma=U+(1-Beta)*K*Y;
        Ds=Upi/(1-exp(-Tvz*10/Td));
    end

    if AutMan==0
        if Rezim==0
            Der1=N*K*((Alfa*W-Y)-Ds+Ds*exp(-Tvz*N/Td));
            Prop=K*(Beta*W-Y);
            U=Prop+Suma+Der1;
            Ds=(Alfa*W-Y)+Ds*exp(-Tvz*N/Td);
            Suma=Suma+K*Tvz/Ti*(W-Y);
        end

        if Rezim==1
            Der1=10*(Upi-Ds+Ds*exp(-Tvz*10/Td));
            Prop=K*(Beta*W-Y);
            Upi=Prop+Suma;
            Ds=(Upi)+Ds*exp(-Tvz*10/Td);
            Suma=Suma+K*Tvz/Ti*(W-Y);

            Der2=(1-exp(-Tvz/(Td/10)))*R_E1+(exp(-vz/(Td/10)))*R_U1;
            R_E1=Upi;
            R_U1=Der2;
```

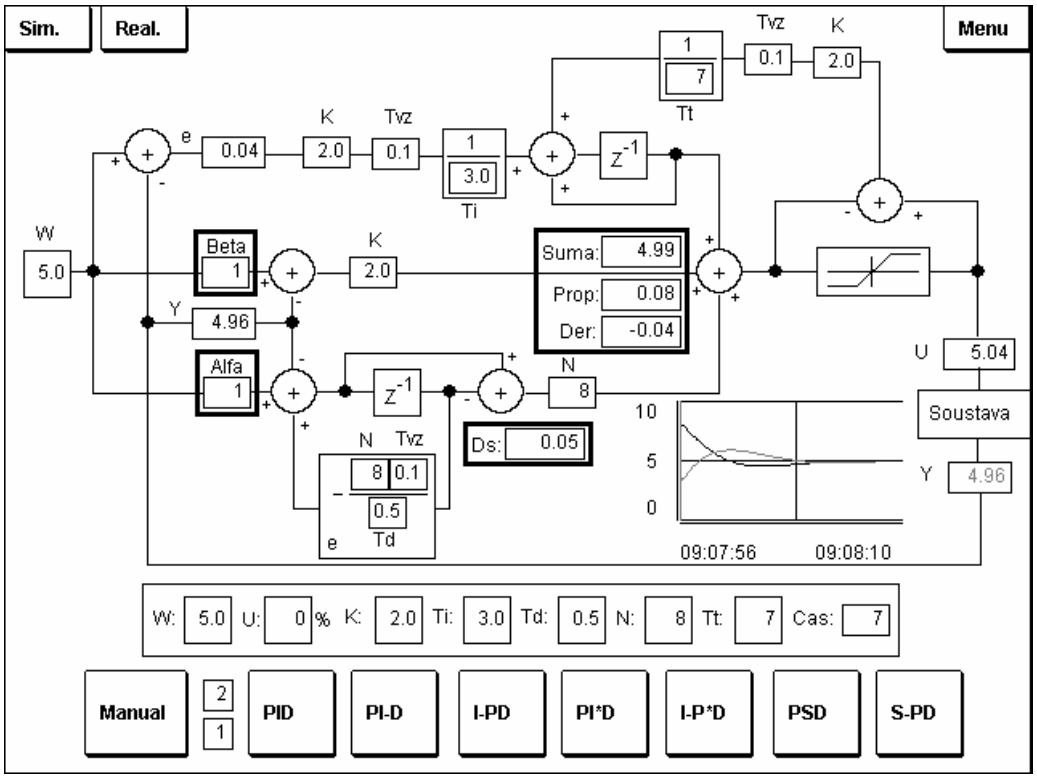
```
    U=Der1+Der2;

    Mat_Der1(i)=Der1;
    Mat_Der2(i)=Der2;
end

if Rezim==2

    if Wpred ~= W   %pokud W neni totozne jako Wpred
        WrozdilCount=2;
    end

    Upred=U;
    R_E0=(W-Y);
    az0=K*(1+Tvz/Ti+Td/Tvz)*R_E0;
    bz1=-K*(1+2*Td/Tvz)*R_E1;
    cz2=K*Td/Tvz*R_E2;
    if (TypPSD==1)
        U=az0+bz1+cz2+R_U1;
    else
        U=az0+bz1+cz2+R_U1+Urozdil;
    end
    R_E2=R_E1;
    R_E1=R_E0;

    if (WrozdilCount>0)&&(TypPSD==1)
        U=Upred;
        WrozdilCount=WrozdilCount-1;
    else
        WrozdilCount=0;
    end

    Mat_Ds(i)=bz1;
    Mat_Prop(i)=cz2;
    Mat_Suma(i)=az0;
end

if U>akc_zas
    if UrozdilCount<3
        Urozdil=U-akc_zas;
        UrozdilCount=UrozdilCount+1;
    else
        Urozdil=0;
    end
    Suma=Suma+(akc_zas-U)*K*Tvz/Tt;
    U=akc_zas;
elseif U<0
    if UrozdilCount<3
        Urozdil=U;
        UrozdilCount=UrozdilCount+1;
    else
        Urozdil=0;
    end
    U=0;
else
    Urozdil=0;
    UrozdilCount=0;
end

R_U1=U;
```

```
        if Suma>int_mez
            Suma=int_mez;
        end
        if Suma<0
            Suma=0;
        end

    end

    if AutMan==1
        W=Y;
        if Rezim == 0
            Suma=U+(1-Beta)*K*Y;
            Ds=Alfa*(W-Y)+(1-Alfa)*(-Y)/(1-exp(-Tvz*N/Td));
        end

        if Rezim == 1
            Upi=U;
            R_U1=U;
            R_E1=U;
            Der1=0;
            Der2=U;
            Suma=U+(1-Beta)*K*Y;
            Ds=Upi/(1-exp(-Tvz*10/Td));
        end

        if Rezim == 2
            R_U1=U;
            R_E0=0;
            R_E1=0;
            R_E2=0;
        end

    end

    if Rezim ~= 2
        Mat_Ds(i)=Ds;
        Mat_Suma(i)=Suma;
        Mat_Prop(i)=Prop;
    end

    Mat_Der1(i)=Der1;
    Mat_Der2(i)=Der2;

    Mat_U(i)=U;
    Mat_Y(i)=Y;
    Mat_Cas(i)=i/10;
    Mat_AutMan(i)=AutMan;

    Y=0.0008105*U1+0.0007883*U2+1.9184012*Y1-0.92*Y2;
    U2=U1;
    U1=U;
    Y2=Y1;
    Y1=Y;
    Y=Y+rand/20000;

    Wpred=W;
End
```

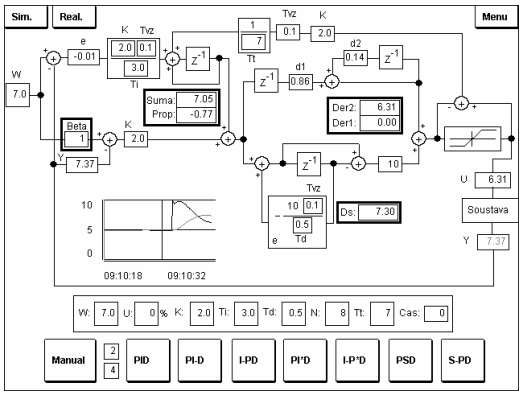
Příloha 2. Popis ovládání vizualizace Power Panelu



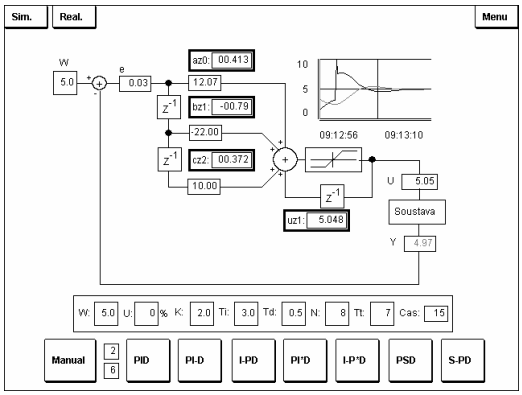
Obr. P2.1: Úvodní obrazovka vizualizace.



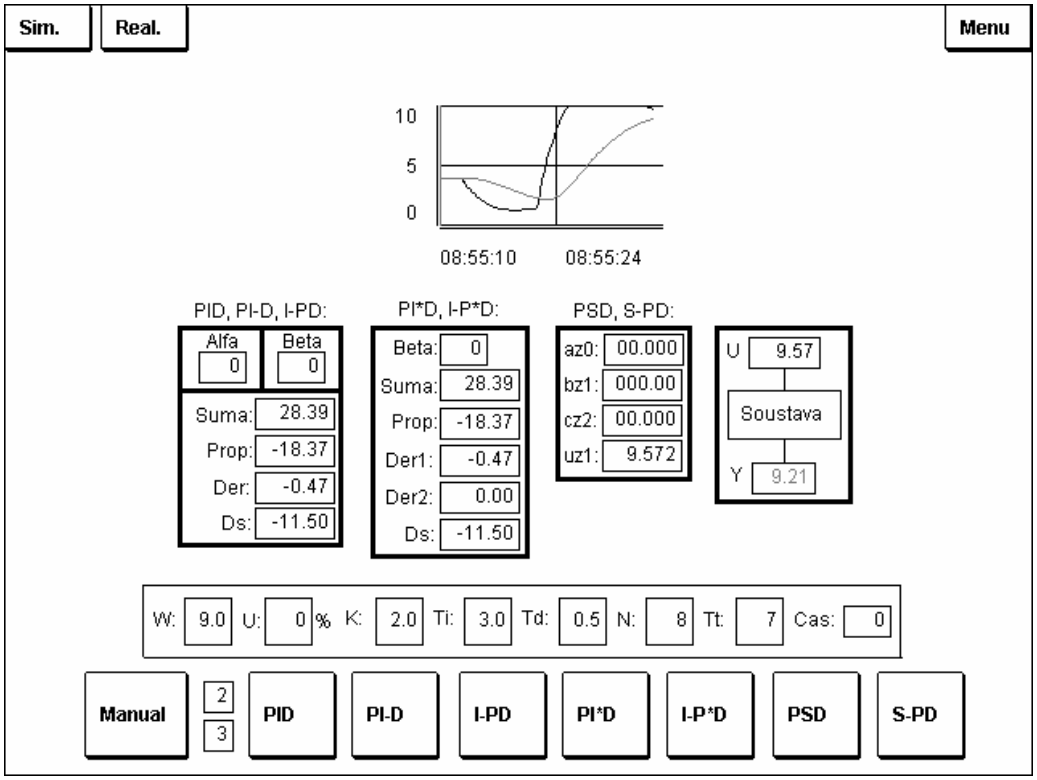
Obr. P2.2: Vizualizace pro PID, PI-D a I-PD regulátory.



Obr. P2.3:Vizualizace pro PI*D var..



Obr. P2.4: Vizualizace pro PSD var..



Obr. P2.5: Zjednodušené zobrazení.

Na úvodní obrazovce si vybereme, jaké zobrazení budeme následně používat. Pokud vybereme grafické zobrazení, budou pro jednotlivé typy algoritmů (zvlášť PID, PI-D a I-PD, zvlášť PI*D a I-P*D, zvlášť PSD, S-PD) vyhrazené jednotlivé obrazovky s úplným zobrazením regulátoru. Nevýhodou tohoto zobrazení je ztráta průběhů v grafu, pokud přepneme mezi jednotlivými obrazovkami. Toto zobrazení je tedy vhodné pro přepínání mezi regulátory jednoho typu. Pokud bychom chtěli přepínat i mezi algoritmy různých typů, vybereme zjednodušení zobrazení. Kromě průběhů jsou zde vypsány hodnoty hlavních složek regulátorů.

Všechny stránky kromě úvodní obsahují jednotné rozhraní pro přepínání algoritmů a nastavování parametrů. V horní nastavovací části můžeme měnit žádanou hodnotu pro automatický režim a uživatelský akční zásah a parametry regulátoru v manuálním režimu. Pokud tyto hodnoty změníme v automatickém režimu, nic se nestane – toto je ošetřeno v programu.

Ve spodní části přepínání regulátorů jsou tlačítka pro přepnutí na manuální režim a na jednotlivé algoritmy regulátorů. Mezi nimi jsou nad sebou umístěná dvě čísla. Horní signalizuje, jestli je aktivní manuální (číslo 1) nebo automatický (číslo 2) režim a spodní udává, který algoritmus je právě aktivní (1 – PID, 2 – PI-D, … , 7 – S-PD). Změna na žádaný algoritmus se provede až když se v kolonce „Cas“ objeví číslo 51 a regulátor označí výstupní hodnotu ze soustavy za ustálenou.

V levém horním rohu obrazovky vidíme přepínač mezi reálnou a simulovanou soustavou. V pravém horním rohu je tlačítko, díky kterého se vrátíme na úvodní obrazovku.

Příloha 3. Cyklický program regulátoru v Power Panelu

```
PROGRAM _CYCLIC
IF SimReal = 0 THEN
      Flow:=REAL_TO_INT(Y*3276);
END_IF
Y:=Flow;
Y:=Y/3276;
IF AutMan=1 THEN
      ZManNaAut:=1;
      IF ZAutNaMan=1 THEN MujAkcniZasah:=REAL_TO_INT(U*10);
            ZAutNaMan:=0;
      END_IF
      U:=MujAkcniZasah*0.1;
      ZadanaAuto:=Y;
END_IF

a:=K*(1+Tvz/Ti+Td/Tvz);
b:=-K*(1+2*Td/Tvz);
c:=K*Td/Tvz;

IF AutMan=2 THEN
      ZAutNaMan:=1;
      W:=ZadanaAuto;
      d1:=(1-EXP(-Tvz/(Td/10)));
      d2:=(EXP(-Tvz/(Td/10)));

      IF ((ZManNaAut=1) OR ((Cas > 50) AND (NovyRez>0))) THEN
            IF ZManNaAut=1 THEN
                  ZManNaAut:=0; Konst[0]:=K;
                  Konst[1]:=Ti; Konst[2]:=Td;
                  Konst[3]:=N; Konst[4]:=Tt;
            END_IF
            IF NovyRez=1 THEN (* PID *)
                  Beta:=1; Alfa:=1;
            END_IF
            IF NovyRez=2 THEN (* PI-D *)
                  Beta:=1; Alfa:=0;
            END_IF
            IF NovyRez=3 THEN (* I-PD *)
                  Beta:=0; Alfa:=0;
            END_IF
            IF NovyRez=4 THEN (* PI*D *)
                  Beta:=1;
            END_IF
            IF NovyRez=5 THEN (* I-P*D *)
                  Beta:=0;
            END_IF
            IF NovyRez=6 THEN (* PSD *)
                  TypPSD:=1;
            END_IF
            IF NovyRez=7 THEN (* S-PD *)
                  TypPSD:=2;
            END_IF
            IF (NovyRez>0) THEN
                  RezimR:=NovyRez; NovyRez:=0;
            END_IF
            IF NovyTypRegulatoru <> TypRegulatoru THEN
```

```
            TypRegulatoru:=NovyTypRegulatoru;
      END_IF

      IF TypRegulatoru = 1 THEN
            Suma:=U+(1-Beta)*K*Y;
        Ds:=Alfa*(W-Y)+(1-Alfa)*(-Y)/(1-EXP(-N*Tvz/Td));
      END_IF
      IF TypRegulatoru = 2 THEN
            Upi:=U;
            R_U1:=U;
            R_E1:=U;
            Suma:=U+(1-Beta)*K*Y;
            Ds:=Upi/(1-EXP(-Tvz*10/Td));
      END_IF
      IF TypRegulatoru = 3 THEN
            R_U1:=U;
            R_E0:=0;
            R_E1:=0;
            R_E2:=0;
      END_IF
ELSE
      K:=Konst[0]; Ti:=Konst[1];
      Td:=Konst[2]; N:=Konst[3];
      Tt:=Konst[4];
END_IF

IF (((Y-W)*(Y-W))<=0.003)AND(SimReal=0) THEN Cas:=Cas+1;
ELSE IF (((Y-W)*(Y-W))<=0.01)AND(SimReal=1) THEN
      Cas:=Cas+1;
      ELSE Cas:=0;
      END_IF
END_IF
IF Cas > 50 THEN
      Cas:=51;
END_IF

IF TypRegulatoru = 1 THEN
      E:=W-Y;
      Der1:=N*K*((Alfa*W-Y)-Ds+Ds*EXP(-N*Tvz/Td));
      Prop:=K*(Beta*W-Y);
      U:=Prop+Suma+Der1;
      Ds:=(Alfa*W-Y)+Ds*EXP(-N*Tvz/Td);
      Suma:=Suma+K*Tvz/Ti*(E);
END_IF
IF TypRegulatoru = 2 THEN
      Der1:=10*(Upi-Ds+Ds*EXP(-Tvz*10/Td));
      Prop:=K*(Beta*W-Y);
      Upi:=Prop+Suma;
      Ds:=(Upi)+Ds*EXP(-Tvz*10/Td);
      Suma:=Suma+K*Tvz/Ti*(W-Y);

      Der2:=d1*R_E1+d2*R_U1;
      R_E1:=Upi;
      R_U1:=Der2;
      U:=Der1+Der2;
END_IF
IF TypRegulatoru = 3 THEN
      IF Wpred <> W THEN
            WrozdilCount:=2;
      END_IF;
```

```
            Upred:=U;
            R_E0:=(W-Y);
            az0:=a*R_E0;
            bz1:=b*R_E1;
            cz2:=c*R_E2;

            IF TypPSD=1 THEN
                  U:=az0+bz1+cz2+R_U1+Urozdil;
            ELSE
                  U:=az0+bz1+cz2+R_U1;
            END_IF

            R_E2:=R_E1;
            R_E1:=R_E0;

            IF ((WrozdilCount>0)AND(TypPSD=2)) THEN
                  U:=Upred; WrozdilCount:=WrozdilCount-1;
            ELSE
                  WrozdilCount:=0;
            END_IF
      END_IF

      IF U>Akc_zas THEN
            IF UrozdilCount<3 THEN
                  Urozdil:=U-Akc_zas;
                  UrozdilCount:=UrozdilCount+1;
            ELSE
                  Urozdil:=0;
            END_IF
            Suma:=Suma+(Akc_zas-U)*K*Tvz/Tt;
            U:=Akc_zas;
      ELSE
            IF U<0 THEN
                  IF UrozdilCount<3 THEN
                        Urozdil:=U;
                        UrozdilCount:=UrozdilCount+1;
                  ELSE
                        Urozdil:=0;
                  END_IF
                  U:=0;
            ELSE
                  Urozdil:=0;
                  UrozdilCount:=0;
            END_IF
      END_IF

      R_U1:=U;

      IF Suma>Int_mez THEN Suma:=Int_mez; END_IF
      IF Suma<0 THEN Suma:=0; END_IF
END_IF

IF SimReal = 0 THEN
      Y:=0.0008105*U1+0.0007883*U2+1.9184012*Y1-0.92*Y2;
      U2:=U1; U1:=U; Y2:=Y1; Y1:=Y;
ELSE
      Fan:=REAL_TO_INT(U*3276);
END_IF

Wpred:=W;
END_PROGRAM
```

Příloha 4. Obsah CD

Kromě elektronické podoby tohoto textu se na přiloženém CD nachází vytvořený m-file s názvem BumplessSwitch.m . Tento skript byl vytvořen a testován v MATLABu R2006a, jinak taky označovaný jako MATLAB v.7.2.0.232. Pro spuštění tohoto skriptu je potřeba zapnout MATLAB, pracovní složku nastavit jako mechaniku CD-ROM a v příkazovém řádku MATLABu napsat text „BumplessSwitch“.

Dalším souborem je program pro virtuální ŘS AR000. Pro jeho spuštění je potřeba nejdříve rozbalit zip soubor s názvem „VirtRS.zip“, ve kterém se tento program nachází, otevřít program Automation Studio v.3 a pomocí příkazu Open Project jej načíst. Následně zapneme samotný AR000 přes hlavní nabídkové menu a záložku Tools. V online nastavení musíme zvolit možnost AR000_TCPIP. Poté program přeložíme a nahrajeme do ŘS. Pro zobrazení vizualizace ŘS zapneme program RealVNC, do úvodní kolonky Server napíšeme 127.0.0.1 a při vyzvání hesla napíšeme 123.

Posledním souborem je vlastní algoritmus do Power Panelu nacházející se v zipu „RealPP.zip“. Stejně jako v minulém případě jej rozbalíme a otevřeme v Automation Studiu v.3. V online nastavení zvolíme TCPIP a v jeho nastavení zvolíme adresu toho Power Panelu, do kterého chceme program nahrát (defaultní ŘS je s adresou 147.229.76.19) a jeho DA. Program přeložíme a nahrajeme do Power Panelu.

Oba tyto programy byly vytvořeny a testovány v Automation Studiu v.3.0.71.10.