TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky
a
mezioborových inženýrských studií

DIPLOMOVÁ PRÁCE

Adaptivní regulátor pro řídicí systém TECO

Liberec 2001

Jan Zelenka

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Katedra řídicí techniky Školní rok: 2000/2001

ZADÁNÍ DIPLOMOVÉ PRÁCE

pro: **Jana Zelenku**

studijní program: 2612T – Elektrotechnika a informatika

obor: Automatické řízení a inženýrská informatika

Vedoucí katedry Vám ve smyslu zákona o vysokých školách č.111/1998 Sb. určuje tuto diplomovou práci:

Název tématu: **Adaptivní regulátor pro řídicí systém TECO**

Zásady pro vypracování:

1. Seznamte se s programovým prostředím řídícího systému TECO
2. Prostudujte identifikační algoritmus průběžného odhadování parametrů metodou LD rozkladu
3. Prostudujte metodu dynamického programování pro návrh adaptivního číslicového regulátoru
4. Naprogramujte adaptivní regulátor ve zvoleném softwarovém prostředí řídícího systému TECO

TECHNICKÁ UNIVERZITA V LIBERCI
Univerzitní knihovna
Voronežská 1329, Liberec 1
PSČ 461 17

Rozsah grafických prací: dle potřeby dokumentace

Rozsah průvodní zprávy: cca 40 až 50 stran

Seznam odborné literatury:

[1] Peterka,V. a kol.: Algoritmy pro adaptivní mikroprocesorovou regulaci technologických procesů.Praha,ÚTIA,ČSAV 1982.

[2] Bobál,V.-Böhm,J.-Prokop,R.-Fessl,J.: Praktické aspekty samočinně se nastavujících regulátorů. VUT Brno, VUTIUM 1999.

[3] Hanuš,B.-Olehla,M.-Modrlák,O.: Číslicová regulace technologických procesů. VUT Brno,VUTIUM 2000.

[4] Janeček,B.: Metoda dynamického programování použitá pro číslicovou regulaci lineárních dynamických soustav. Habilitační práce, TU Liberec,1996.

Vedoucí diplomové práce: Doc.Ing.Osvald Modrlák,CSc.

Konzultant: Doc.Ing.Bedřich Janeček,CSc.

Zadání diplomové práce: 25.10.2000

Termín odevzdání diplomové práce: 25. 5. 2001

L.S.

Technická univerzita v Liberci * Fakulta mechatroniky a mezioborových inženýrských studií

.....

Vedoucí katedry

.....

Děkan

V Liberci dne 25.10.2000

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: 2612T Elektrotechnika a informatika

Studijní obor: Automatické řízení a inženýrská informatika

Adaptivní regulátor pro řídicí systém TECO

(Adaptive controller for control system TECO)

Jan Zelenka

Vedoucí práce: Doc.Ing.Osvald Modrlák,CSc.
Technická univerzita Liberec

Konzultant: Doc.Ing.Bedřich Janeček,CSc.
Technická univerzita Liberec

Rozsah práce a příloh:

Počet stran textu: 82

Počet obrázků: 54

Počet tabulek : 6

Počet vzorců : 52

Počet příloh : 4

Anotace:

Tato diplomová práce se zabývá adaptivním přístupem řízení technologických procesů, konkrétně návrhem adaptivního řídicího systému s průběžnou identifikací. Pro návrh takovéhoto systému jsou použity účinné algoritmy umožňující realizovat adaptivní regulátor malým průmyslovým počítačem, v tomto případě PLC automatem firmy TECO a.s., vyráběného pod ochranou známkou TECOMAT. Pro identifikaci lineární dynamické soustavy je použit algoritmus průběžného odhadování parametrů metodou LD rozkladu a pro návrh číslicového regulátoru je použita metoda dynamického programování.

Abstract:

The Diploma Thesis deal progressive methods of adaptive control of dynamic systems. The goal of Thesis is to design adaptive controller realized by Programmable Logic Controller TECOMAT by TECO a.s. For design of that adaptive controller (self tuning controller) are developed effective methods, which allow realize adaptive controller by industrial computers. For identification of linear dynamic system is used algorithm of survey of parameters by method of lower diagonal decomposition. The synthesis of control is designed by method of dynamic programming used for digital control.

Poděkování:

Na tomto místě bych chtěl poděkovat Doc. Ing. O. Modrlákovi, CSc. za odborné vedení, pomoc při zpracování diplomové práce, za cenné rady a poskytnuté informace. Dále mé poděkování patří konzultantovi Doc. Ing. B. Janečkovi, CSc. za poskytnuté připomínky a rady.

Prohlášení :

„Místopřísežně prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury."

V Liberci dne 24. 5. 2001 Jan Zelenka

Prohlášení k využívání výsledků diplomové práce :

Jsem si vědom toho, že diplomová práce je majetkem školy s že s ní nemohu sám bez svolení školy disponovat, že diplomová práce může být zapůjčena, či objednána (kopie) za účelem využití jejího obsahu.

Beru na vědomí, že po pěti letech si mohu diplomovou práci vyžádat v Univerzitní knihovně TU v Liberci, kde je uložena.

Jan Zelenka
Mozartova 669/17, Liberec 1, 460 01

Obsah:

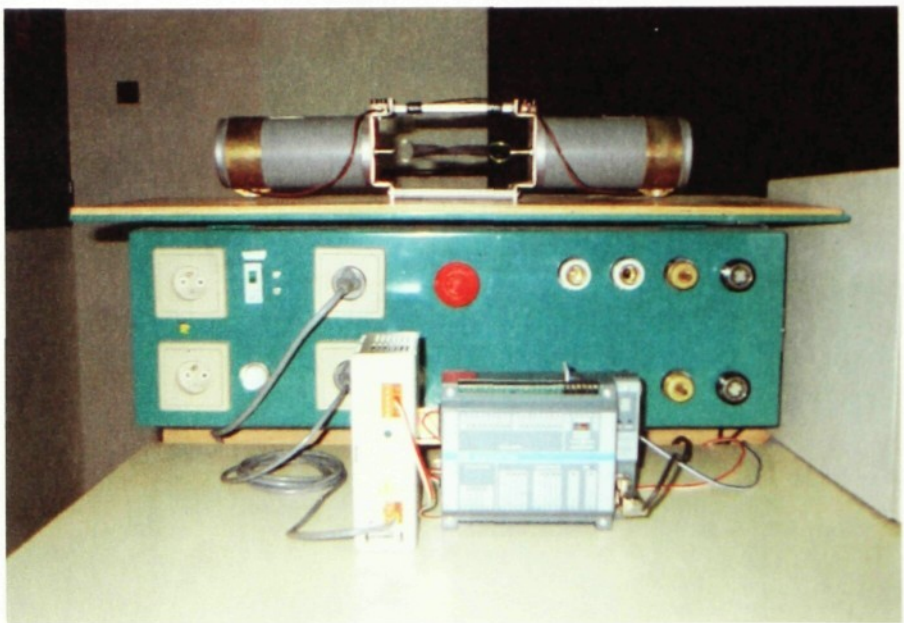
1. Úvod

V zásadě dnes existují dva různé moderní přístupy v regulaci. Je to robustnost a adaptibilita. Regulace při pomalých změnách parametrů regulované soustavy se právě řeší v teoretické oblasti v disciplíně robustní a adaptivní řízení. Adaptivní regulace je jedním z možných vyšších způsobů řízení. Pod pojmem vyšší způsob řízení lze chápat takové řízení, které splňuje vyšší požadavky na kvalitu řízení, přizpůsobivost řízení měnícím se podmínkám nebo dokonce schopnost řídicího systému využívat historie řídících procesů ke svému učení. Mimo jiné i touto problematikou se zabývají na Katedře řídicí techniky TU v Liberci. Tedy tato činnost byla podnětem k zadání diplomové práce, která se tématicky zabývá adaptivním přístupem v regulaci, konkrétně jde o návrh adaptivního řídicího systému s průběžnou identifikací. V takovýchto systémech je zabudován algoritmus průběžné identifikace, který odhaduje a upřesňuje parametry regulované soustavy. Pak podle povahy řízeného procesu lze sledovat splnění těchto cílů: automatické seřízení číslicového regulátoru, zachycení změn parametrů řízené soustavy a následné zlepšení regulačních pochodů vhodnou změnou parametrů číslicového regulátoru.

Takovéto číslicové regulátory pracují s časovou periodou, která je podstatně delší než perioda činnosti spojitých regulátorů, realizovaných na číslicových prvcích. Z hlediska periody činnosti je ověřeno, že číslicové regulátory nezhoršují prakticky kvalitu regulace ve srovnání se spojitými regulátory. Číslicový regulátor je možné realizovat jako regulátor stejného řádu jako je řád regulované soustavy. Z toho důvodu u soustav vyšších řádů než je řád dvě dosáhneme použitím číslicových regulátorů vyšší kvality regulace než při použití spojitých PID regulátorů. Pro návrh číslicových regulátorů je tedy nutné určit - identifikovat diskrétní model regulované soustavy. Pro identifikaci těchto modelů a pro návrh číslicových regulátorů jsou vyvinuty účinné rekurzivní algoritmy. Jedná se o numericky stabilní metody, s malými požadavky na operační paměť počítačů. Díky těmto vlastnostem jsou vhodné pro stavbu adaptivních regulátorů realizovaných i na malých průmyslových řídicích systémech.

Cílem diplomové práce je vytvořit adaptivní regulátor realizovaný PLC automatem TECOMAT firmy TECO a.s. a odzkoušet jeho vlastnosti na dynamické

soustavě. Pro jeho návrh jsou použity vhodné rekurzivní algoritmy, konkrétně pro identifikaci je použit algoritmus průběžného odhadování parametrů metodou LD rozkladu a pro syntézu - návrh číslicového regulátoru je použita metoda dynamického programování.



Obr.1.1 Pracoviště v laboratoři Katedry řídicí techniky

2. PLC Tecomat a programové prostředí Epos for Windows

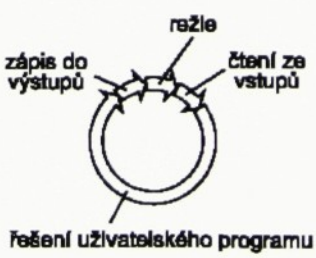
2.1. Funkce, vlastnosti a parametry PLC

V této kapitole bych se rád zmínil o funkcích a vlastnostech programovatelných logických automatů TECOMAT. Vzhledem k tomu, že jsem během studia neměl možnost pracovat s těmito PLC automaty, musel jsem se poměrně důkladně seznámit s jejich činností a vlastnostmi. Informace jsem čerpal především z dostupných uživatelských manuálů [7],[8] a katalogů [9], [10]. Důležitým zdrojem byly i www stránky firmy TECO a.s.

2.1.1. Princip vykonávání uživatelského programu

Programovatelný logický automat (dále jen PLC - Programmable Logic Controller) je číslicový elektronický řídicí systém určený pro řízení procesů v průmyslovém prostředí. Používá programovatelnou paměť pro vnitřní ukládání uživatelsky orientovaných instrukcí, jež slouží k realizaci specifických funkcí pro řízení různých druhů strojů nebo procesů prostřednictvím číslicových nebo analogových vstupů a výstupů.

Řídící algoritmus PLC je zapsán jako posloupnost instrukcí v paměti uživatelského programu. Centrální jednotka (dále jen CPU – Central Procesor Unit) postupně čte z této paměti jednotlivé instrukce, provádí příslušné operace s daty v zápisníkové paměti a zásobníku. Jsou-li provedeny všechny instrukce požadovaného algoritmu, provádí CPU aktualizaci výstupních proměnných do výstupních periferních jednotek a aktualizuje stavy ze vstupních periferních jednotek do zápisníkové paměti. Tento děj se stále opakuje a nazýváme jej cyklem programu (obr.2.1). Činnosti zápisu do výstupů, režie a čtení ze vstupů jsou souhrnně nazývány otočka cyklu.

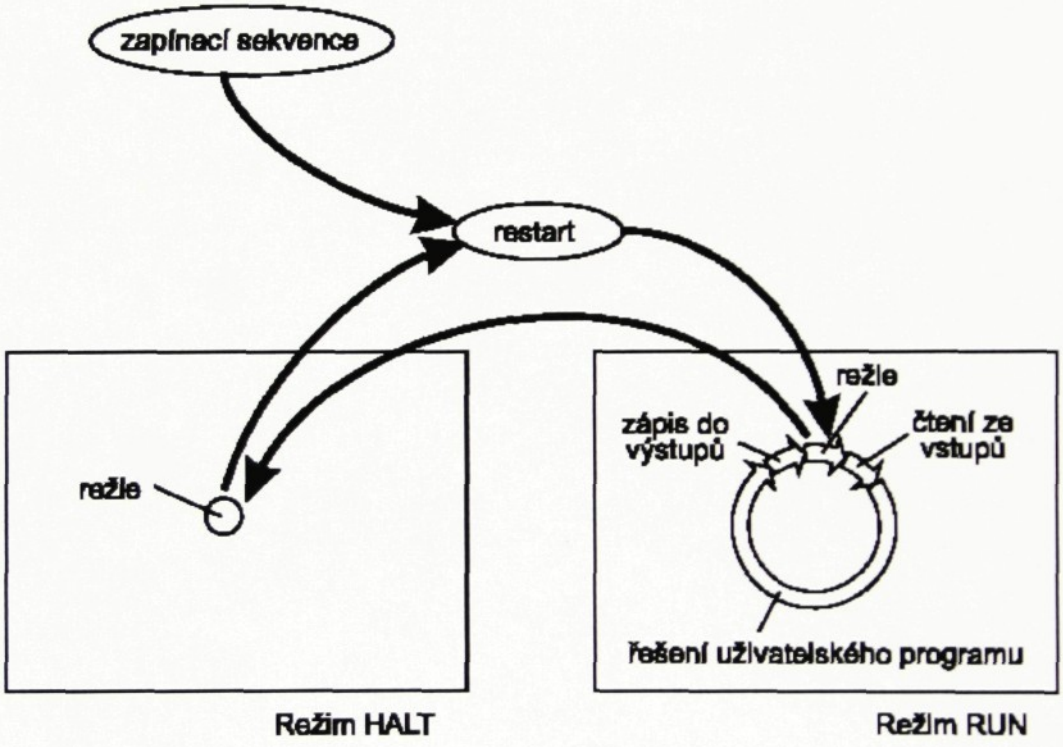


Obr.2.1 Cyklus programu

Jednorázová aktualizace stavů vstupních proměnných během celého cyklu programu odstraňuje možnosti vzniku hazardních stavů při řešení algoritmu řízení (během výpočtu nemůže dojít ke změně vstupních proměnných).

Před zahájením psaní vlastního uživatelského programu pro PLC je třeba si tuto skutečnost uvědomit. V některých případech to usnadňuje řešení problému, v jiných zase komplikuje.

Zjednodušené schéma řešení uživatelského programu v PLC je uvedeno na obr.2.2. Jednotlivé pojmy jsou vysvětleny v následujících kapitolách.



Obr.2.2 Řešení uživatelského programu v PLC

Zapínací sekvence:

Zapínací sekvencí rozumíme činnost PLC bezprostředně po zapnutí napájení. Obsahuje otestování sw i hw PLC a nastavení PLC do definovaného výchozího stavu, nebo vyvolání nastavovacího režimu určeného pro zadání výchozích parametrů.

Po ukončení zapínací sekvence je proveden restart, PLC přejde do režimu RUN a začne vykonávat uživatelský program. Pokud během zapínací sekvence diagnostika PLC vyhodnotila kritickou chybu, zůstává PLC v režimu HALT a signalizuje chybu.

Pokud byl vyvolán nastavovací režim, po jeho ukončení proběhne zapínací sekvence, ale pak PLC přejde do režimu HALT, uživatelský program není vykonáván, výstupy PLC zůstávají zablokované a PLC očekává příkazy z nadřízeného systému. Uživatelský program lze spustit buď pomocí nadřízeného systému, nebo vypnutím a zapnutím napájení.

Pracovní režimy PLC:

PLC může pracovat ve dvou základních režimech. Tyto pracovní režimy jsou označeny RUN a HALT.

V režimu RUN načítá PLC hodnoty vstupních signálů, řeší instrukce uživatelského programu a zapisuje vypočtené hodnoty výstupních signálů. Jedno provedení těchto činností představuje cyklus programu. Čtení ze vstupů představuje přepis hodnot ze vstupních jednotek PLC do oblasti X v zápisníkové paměti a zápis do výstupů představuje přepis hodnot vypočtených uživatelským programem z oblasti Y zápisníkové paměti do výstupních jednotek PLC.

Z obr.2.2 vyplývá, že v případě PLC jde o nespojité vyhodnocování vstupů (obecná vlastnost odlišující digitální systémy od plně analogových), jehož vzorkovací frekvence je daná dobou cyklu, která je určená především velikostí a strukturou uživatelského programu. Doba cyklu se pohybuje podle výkonu centrální jednotky řádově od jednotek přes desítky až stovky milisekund. Tato vlastnost je nevýhodou pro danou aplikaci PLC, kdy je potřebné zajistit pevně danou a předem známou vzorkovací frekvenci. Dalším požadavkem je dosažení celkem nízké hodnoty doby cyklu (řádově 100ms), při poměrně rozsáhlém algoritmu výpočtu, která zaručí možnost řízení rychlé dynamické soustavy.

Režim HALT slouží především k činnostem spojeným s edicí uživatelského programu. V tomto režimu není program vykonáván a není ani prováděn přenos dat mezi CPU a periferiemi.

Změnu pracovních režimů PLC lze provádět pomocí nadřízeného systému. Obecně platí, že změna pracovního režimu PLC je činnost vyžadující zvýšenou pozornost obsluhy, neboť v mnoha případech velice výrazně ovlivňuje stav řízeného objektu. Příkladem může být přechod z režimu RUN do režimu HALT, kdy PLC přestane řešit uživatelský program a připojený objekt přestává být řízen.

Restarty uživatelského programu:

Restartem se rozumí taková činnost PLC, jejímž úkolem je připravit PLC na řešení uživatelského programu. Restart se za normálních okolností provádí po úspěšném zakončení zapínací sekvence a při každé změně uživatelského programu. PLC TECOMAT rozlišují dva druhy restartu, teplý a studený. Teplý restart umožňuje zachování hodnot v registrech i během vypnutého napájení (pouze remanentní zóna). Studený restart provádí vždy plnou inicializaci paměti. Uživatelský program je také možné spustit bez restartu, v tomto případě se provádí pouze test neporušenosti uživatelského programu a kontrola periferního systému PLC.

V závislosti na prováděném restartu pracuje také plánovač uživatelských procesů. Při teplém restartu je jako první řešen uživatelský proces P62, při studeném restartu je jako první po přechodu do RUN řešen proces P63. Není-li restart při přechodu do RUN prováděn, řeší se jako první proces P0.

2.1.2. Uživatelské procesy

Uživatelský program se skládá z uživatelských procesů. Teoreticky jich smí být až 65 (0 až 64), prakticky jich bývá výrazně méně. Procesy jsou aktivovány podle předem daných pravidel. Lze tedy ovlivnit aktivaci většiny procesů v průběhu uživatelského programu.

Přehled všech procesů uživatelského programu a jejich určení obsahuje tab.2.1.

Tab.2.1 Procesy uživatelského programu a jejich určení

Procesy	Určení
P0	základní proces
P1 až P4	čtyřfázově aktivované procesy
P5 až P9	časově aktivované procesy
P10 až P40	uživatelsky aktivované procesy
P41 až P48	přerušovací procesy
P50 až P57	ošetření ladicího bodu
P60	balík podprogramů
P62	teplý restart
P63	studený restart
P64	závěrečný proces cyklu

Uživatel není nucen využívat všechny procesy. Pokud mu vyhovuje klasické jednosmyčkové řízení, může zadat pouze proces P0. Procesy lze vyloučit trojím způsobem. Buď proces není vůbec naprogramován. Pouze proces P0 nelze takto vyloučit, ale musí být uveden. Nebo daný proces je prázdný. Jeho aktivace se projeví pouze jako prázdná operace. A do třetice, aktivační maska procesu je vynulovaná. Aktivační masky procesů P10 až P48 jsou obsaženy v systémových registrech. Proces s vynulovanou aktivační maskou bude potlačen v následujícím cyklu, respektive ihned, jedná-li se o přerušující proces P40 až P48. Systémem spravované procesy P0 až P9 nelze takto potlačit. Tedy procesy P0 a P64 se aktivují v každém cyklu, procesy P1 až P9 se aktivují ve vybraných cyklech, procesy P10 až P40 aktivuje uživatel pomocí řídících masek. Výsledně se tyto procesy jeví jako různé smyčky uživatelského programu. Proto tento způsob aktivace lze označit jako vícesmyčkové řízení.

Ošetření restartu, procesy P62 a P63:

V prvém cyklu po restartu může být aktivován jeden z procesů P62, P63. Tyto procesy slouží k inicializaci proměnných. Pokud je prováděn jeden z procesů P62 nebo P63, není již v tomto cyklu prováděn žádný jiný proces, tedy ani P0.

Pokud je nastaven teplý restart, provede se proces P62 a v následujícím cyklu se začne provádět proces P0 a další naprogramované procesy. Není-li naprogramován proces P62, provede se místo něj proces P63. Pokud není naprogramován ani jeden z procesů P62, P63, začne se rovnou provádět proces P0 a další naprogramované procesy.

Je-li nastaven studený restart, provede se proces P63 a ostatní je analogické jako v předchozím případě.

Pokud je nastaven režim bez restartu, neprovede se ani jeden z procesů P62, P63, ale začne se opět rovnou provádět proces P0 a další naprogramované procesy.

Základní proces P0:

Proces P0 je úvodní proces každého cyklu. Je povinnou součástí základní struktury uživatelského programu. Je účelné sem zařadit všechny úvodní operace a uživatelský plánovač procesů.

Čtyřfázově aktivované procesy P1, P2, P3, P4:

Procesy se v aktivaci cyklicky střídají v pořadí P1,P2 P3, P4, P1, ... Jejich základní vlastností je, že v každém cyklu je aktivní právě jeden. To je umožňuje použít pro programování kolizních akcí, které se nesmějí provádět ve stejném cyklu.

Časově aktivované procesy P5, P6, P7, P8, P9:

Tyto procesy jsou aktivovány vždy po uplynutí určitého časového intervalu. Přesnost tohoto intervalu je dána dobou cyklu. Jednotlivé intervaly jsou 400ms 3,2s 25,6s 3,4min 27,2min. Použití procesů je vhodné zejména v případech typu: „ několikrát za sekundu proveď..." , „ po několika sekundách proveď..." , atd.

Uživatelsky aktivované procesy P10 až P40:

Tyto procesy jsou aktivovány uživatelem nastavením řídících bitů v systémových registrech S25 až S28. Nastavením příslušného bitu na log.1 se v následujícím cyklu aktivuje příslušný proces.

Přerušující procesy P41 až P48:

Kterýkoliv z procesů smyčky může být přerušen některým z přerušujících procesů. Systémový program zajistí dokončení právě rozpracované instrukce, odloží stav aktivního zásobníku a předá řízení na začátek přerušujícího procesu. Jeho závěrečná instrukce vrací nepoškozený stav zásobníku a vrátí řízení přerušenému procesu na místo přerušení. Přerušující proces již nemůže být přerušen (není možné vnořování přerušení).

2.1.3. Struktura zápisníkové paměti

Zápisníkem nebo též zápisníkovou pamětí rozumíme úsek paměťového prostoru PLC, který je přístupný jak pro čtení, tak i pro zápis uživatelských dat. Instrukce PLC umožňují přístup na libovolnou část zápisníku. Tato paměť je předem rozdělena do jednotlivých částí s předem vyhrazeným významem. Schematicky je uspořádání zápisníkové paměti zobrazeno na obr.2.3.

Řada centrálních jednotek	E	M	S	D	B
Obrazy vstupních signálů **X**	X0 : X15	X0 : X15	X0 : X127	X0 : X127	X0 : X127
Obrazy výstupních signálů **Y**	Y0 : Y15	Y0 : Y15	Y0 : Y127	Y0 : Y127	Y0 : Y127
Systémové registry **S**	S0 : S63	S0 : S63	S0 : S63	S0 : S63	S0 : S63
Uživatelské registry **R**	R0 : R255	R0 : R255	R0 : R511	R0 : R8191	R0 : R8191

Obr.2.3 Struktura zápisníkové paměti včetně rozsahů operandů pro jednotlivé řady centrálních jednotek

Zápisníková paměť je rozdělena na tyto části: obrazy vstupních signálů X, obrazy výstupních signálů Y, systémové registry S, uživatelské registry R.

Všeobecně je dodržována zásada, že přístup systémového programu k zápisníkové paměti se výhradně uskutečňuje ve fázi otočky cyklu uživatelského programu. To se týká nejenom snímání fyzických vstupů do oblasti X a nastavení hodnot z oblasti Y na fyzické výstupy, ale i změn systémových proměnných S. To znamená, že po dobu cyklu uživatelského programu jsou data v zápisníku zmražena a aktualizují se až po nejbližší otočce cyklu.

2.1.4. Parametry PLC TECOMAT TC500

Pro realizaci dané aplikace jsem měl možnost použít PLC řady TC500, konkrétně typ TC515. Přestože jsou PLC řady TC500 určeny pro nejmenší aplikace, zůstávají zachovány užitné vlastnosti velkých PLC TECOMAT. Významnou vlastností je jednotnost technických a programových prostředků pro tvorbu a ladění uživatelského programu.

Přehled parametrů:

Jednotlivé základní parametry jsou přehledně uspořádány do tabulek. V tab.2.2 jsou uvedeny základní parametry PLC řady TC500.

Tab.2.2 Základní parametry

Druh zařízení	vestavné
Třída elektrického předmětu	I dle ČSN 33 0600
Krytí	IP-10B, čelní panel IP-54
Napájecí napětí (SELV)	24 V~ ±20 %, 50-60 Hz ±5 % nebo 24 V- ±20 %
Příkon	max. 20 VA nebo 13 W
Hmotnost	cca 1 kg
Rozměry	172x192x76 (vxšxh)

Centrální jednotka:

Centrální jednotka zajišťuje většinu řídících funkcí PLC. Řada TC500 je osazena CPU řady D. Obsahuje především měnič napájecího napětí, mikrořadič, paměti RAM a EEPROM, obvod RTC, lithiovou baterii pro napájení paměti dat a obvodu RTC při vypnutí napájení PLC, dva sériové komunikační kanály a analogové výstupní obvody. Základní parametry CPU jsou uvedeny v tab.2.3.

Paměť uživatelského programu je část paměti RAM CPU, vyhrazená pro uživatelský program, data a tabulky.

Zdrojová paměť uživatelského programu je část paměti EEPROM CPU, vyhrazená pro uložení kopie uživatelského programu. Paměť je energeticky nezávislá, tedy obsah paměti zůstává zachován i po vypnutí napájení, nebo při vybité baterii. Funkce slouží především k záloze uživatelského programu.

Zápisníková paměť je část paměti RAM CPU, dostupná uživateli jako obrazy vstupů (registry X), obrazy výstupů (registry Y), systémové (S) a uživatelské registry (R).

Tab. 2.3 Základní parametry CPU

Řada centrální jednotky	D
Obvod reálného času (RTC)	standardně osazen
Zdrojová paměť uživatelského programu	standardně osazena
Druh paměti	EEPROM (FLASH)
Velikost paměti	32 kB
Paměť uživatelského programu a dat	standardně osazena
Druh paměti	RAM
Velikost paměti	32 kB
Přídavná paměť dat; DataBox	volitelná
Druh paměti	RAM
Velikost paměti	128 kB nebo 512 kB
Zálohování pamětí RAM a RTC	min. 20 000 h
Doba cyklu na 1k logických instrukcí	13 ms
Celkový počet uživatelských registrů	8 192
Počet remanentních registrů	volitelný 0 až 512
Celkový počet časovačů a čítačů	4 096
Rozsah časovačů	65 536 x 10 ms až 10 s, možnost kaskádování
Rozsah čítačů	65 536, možnost kaskádování
Instrukční soubor	rozšířený
Délka instrukce	1 až 6 bytů
Počet sériových komunikačních kanálů	2
Přenosová rychlost CH1	0,3 až 57,6 kBd
Přenosová rychlost CH2	0,3 až 230,4 kBd
Počet analogových výstupů	alternativně 0 nebo 4

Jednotka vstupů a výstupů:

Na jednotce vstupů a výstupů je realizována většina vstupních a výstupních obvodů PLC. Jednotlivé typy řady se liší modifikací osazení jednotky obvody binárních a analogových vstupů a binárních tranzistorových, reléových a analogových výstupů.

Pro potřeby dané aplikace PLC jsem používal pouze analogové vstupy a výstupy, uvádím zde tedy jen příslušné základní parametry.

Analogové vstupy slouží k připojení analogových signálů řízeného objektu k PLC. Vstupy jsou uspořádány do skupiny s jednou společnou svorkou analogové

země. Analogové výstupy slouží k připojení napěťově řízených akčních prvků řízeného objektu. Jsou také uspořádány do skupiny se společnou svorkou analogové země.

Nedílnou součástí uživatelského programu je softwarová konfigurace vstupů a výstupů. Zadává se pomocí direktivy *#unit*, kde je nutné zadat typ vstupů a výstupů, počet vstupních bytů, počet výstupních bytů, umístění prvního vstupního a výstupního bytu v zápisníku a jejich aktivaci. Tato deklarace představuje obsluhu a nastavení jednotky vstupů a výstupů.

V tab.2.4 a 2.5 jsou uvedeny základní parametry analogových vstupů a výstupů.

Tab.2.4 Parametry analogových vstupů.

	TC501 až TC504 TC511 až TC514	TC505, TC506 TC515, TC516
Počet vstupních kanálů	-	4
Uspořádání (počet skupin x počet vstupů)		1x4
Společný vodič skupiny	minus	
Galvanické oddělení od interních elektrických obvodů	ne	
Binární reprezentace vstupu	12 bitů bez znaménka	
Doba převodu 4 kanálů	4 ms	
Typ vstupu	proudový nebo napěťový	
Napěťové vstupy		
Měřicí rozsah/ rozlišení (1 LSB)	0 V až +10 V/ &2,44 mV [1] 0 V až +2 V/ &0,49 mV [2]	
Chyba plného rozsahu	max. 0,4%	
Vstupní odpor	>10 MΩ	
Doporučený vnitřní odpor zdroje signálu	<10 kΩ	
Proudové vstupy		
Měřicí rozsah/ rozlišení (1 LSB)	0 mA až +20 mA/ &4,9 µA	
Vstupní odpor	100 Ω	
Vstupní proud	max. 50 mA	
Vstupní napětí	max. 5 V	

Tab.2.5 Parametry analogových výstupů

	TC501 až TC506	TC511 až TC516
Počet výstupních kanálů	-	4
Uspořádání výstupů		1x4
Společný vodič skupiny		minus
Galvanické oddělení od interních elektrických obvodů		ne
Typ výstupu		napěťový
Napěťový rozsah		&0 až 9,96 V
Rozlišení (1 LSB [1)])		&39 mV
Chyba výstupního napětí		typ. ± 1 LSB max. ± 4 LSB
Binární reprezentace výstupu		8 bitů
Výstupní proud		max. 10 mA
Doba nastavení výstupu		max. 30 µs
Zatěžovací odpor výstupu		>1 kΩ
Odolnost proti zkratu		min. 5 s

Ovládací panel:

Ovládací panel je určen především k zadávání parametrů uživatelského programu a zobrazování důležitých provozních stavů vyhodnocených uživatelským programem. Kromě toho je také využíván systémem k zobrazení diagnostických hlášení.

2.2. Programové prostředí EPOS for Windows

Programování řídících algoritmů (uživatelského programu) a testování správnosti napsaných programů pro PLC TECOMAT se provádí v takzvaném nadřízeném systému, počítači standardu PC. Pro spojení s PLC se využívá běžný sériový kanál těchto počítačů.

Firma TECO a.s. nabízí dvě různá programová prostředí umožňující tvorbu aplikací pro řídicí systémy TECOMAT. Jedním je vývojové prostředí xPRO a druhým Epos for Windows. Zásadním rozdílem je operační systém pod kterým tyto prostředí pracují a způsob programování. Programové prostředí xPRO pracuje pod operačním systémem DOS a vývoj aplikace se provádí buď programováním reléových schémat

nebo pomocí instrukčního souboru. Prostředí Epos pracuje pod platformou Windows a navrhovaná aplikace se programuje ve strukturovaném jazyku. Jelikož jsem neměl zkušenosti s programováním těchto PLC automatů přímo prostřednictvím mnemokódu, tedy pomocí dostupného instrukčního souboru, volil jsem možnost programovat ve vyšším strukturovaném jazyku. Z tohoto důvodu jsem za vývojové prostředí zvolil software Epos for Windows. Toto programové prostředí a syntaxe jazyka JLR je důkladně popsána v [12] a [13], proto zde uvádím jen stručnou charakteristiku.

2.2.1. Základní vlastnosti a funkce

Epos for Windows je plně 32-bitový programovací prostředek pro vývoj aplikací pro všechny druhy PLC firmy TECO a.s. Pracuje pod operačním systémem Windows 95/NT. Je určen pro návrh, ladění a údržbu programů. Lze ho použít pro návrh aplikace jak pro jeden PLC, tak i pro tvorbu projektů s velkým počtem automatů. Epos používá vyšší programovací jazyk Pascalského typu s možností průběžného použití mnemonických instrukcí. Samotné ladění programu může probíhat ve dvou režimech, on-line a off-line. V prvním případě je k počítači PC připojen přes sériovou linku skutečný automat, v druhém je automat simulován přímo v PC. Další funkcí Eposu je spolupráce se systémovým analyzátorem PLC (program Analyzer) a vytváření uživatelských rozhraní operátorských panelů (program Panel Tool).

K dispozici jsem měl volně šiřitelnou Lite verzi prostředí Epos for Windows 1.3. Prostředí je plně funkční, omezení je v délce kódu programu na 4 kB. Po připojení hardwarového klíče do paralelního portu PC není délka kódu programu omezená a její maximální velikost je 32 kB. Nevýhodou Lite verze je, že při ladění programu v off-line režimu, tedy na simulovaném PLC, toto omezení platí také. Pokud jsem chtěl tedy pracovat na mém programu doma, kde jsem neměl k dispozici hardwarový klíč, nemohl jsem spustit režim ladění, přestože celé zařízení PLC mohlo být simulováno v PC. Výhodou spuštění ladění programu nejprve v Off-line režimu a poté teprve skutečné natažení programu do PLC v on-line režimu, je možnost krokování jednotlivých instrukcí, nebo simulace cyklického zpracování programu. Dále je to také sledování hodnot daných proměnných.

Prostředí Epos for Windows používá programovací jazyk nazvaný JLR. Jeho označení jako vyšší programovací jazyk Pascalského typu se mi zdá příliš nadnesené. Je zde sice zachována obecná struktura definic a deklarací, použití maker odpovídající pascalským funkcím a procedurám, ale zápis strukturovaného programu příkazy typu for, while, goto má jistá omezení. Je to například použití cyklu for, kde není možné vnořování příkazů cyklu, nebo není možný zápis příkazu for do těla makra. Dále je to způsob indexování strukturovaných proměnných typu array, kdy se na danou hodnotu vektoru nedá odkazovat obyčejně indexem (pomocí proměnné, např. *Da[k]*), ale jen klíčovým slovem „*i*" použitým v cyklu for. Při naprogramování vnořených cyklů pomocí příkazu *goto*, *návěští* a podmínky *if*, kdy jsem si sám vypočítával indexy jednotlivých vektorů, ale narazil jsem na výše uvedené omezení a zjistil jsem, že se takto odkazovat na jednotlivé prvky nelze. Podle autora Eposu se problém se zápisem cyklu for ve for se stále řeší. Jednoduché řešení neexistuje - PLC NS950 nepodporuje indexování (částečně pro bool, byte a word pomocí instrukce LTB a WTB), takže volání indexovaných proměnných se musí provádět přes okliku pomocí instrukcí SRC a MOV.

Ve verzi Epos for Windows 1.3 je možné pracovat se strukturovanými proměnnými, je plně podporována práce s čísly s plovoucí řádovou čárkou nebo je zde možnost přímého použití mnemoinstrukcí v programu. Důležitou vlastností je možnost psát programy ve vyšším jazyku, bez znalosti instrukčního souboru.

2.2.2. Spolupráce se systémovým analyzátorem

Program Epos umožňuje přímou práci s tímto analyzátorem [14]. Ve skutečnosti se jedná o další program, Analyzer for Windows v2.0. V případě systémového analyzátoru programovatelného automatu se v podstatě jedná o paměťový buffer, do kterého se v určených časových intervalech zapisují hodnoty z vybraných adres. Program Analyzer for Windows umožňuje vyčtení těchto hodnot z PLC do počítače a dále jejich jednoduché zpracování. Je určen především tvůrcům aplikace při ladění programu a pro servisní účely. Délka paměťového buffru je 6 kB. Maximální počet sledovaných proměnných je 16, proměnné mohou být typu bit, byte a word. Není tedy podporován typ float. Záleží tedy na parametrech nastavení (jakého typu budou proměnné, kolik jich bude a jak často se budou vzorkovat), jak skutečně dlouhý časový

interval lze zobrazit. Začátek vzorkování proměnných je podmíněn takzvaným signálem trigg. Je to v podstatě logický signál generovaný na základě podmínek přiřazeným k jednotlivým snímaným proměnným. Tímto signálem se řídí zápis hodnot do bufferu.

V mém případě se jednalo o zobrazení průběhů výstupních otáček regulované soustavy, zidentifikovaného diskrétního modelu soustavy a dále pak regulačních pochodů. Podle zadaného intervalu vzorkování daných proměnných se možná doba nasnímaných průběhů pohybovala od jednotek až po desítky sekund.

2.2.3. *Programování ovládacích panelů*

Aplikaci operátorských panelů ID-04, ID-05 a panelu PLC TC500 zajišťuje Epos pomocí grafického návrháře uživatelských rozhraní, PanelTool. Program využívá vlastností uživatelské instrukce TER, která byla právě vytvořena pro podporu programování displejů ovládacích panelů. Prostředí umožňuje zadávání textů, umístění proměnných, definice menu a uživatelských zpráv. Výsledkem návrhu jsou vygenerované, přeložitelné soubory, které se připojují ke zdrojovému programu.

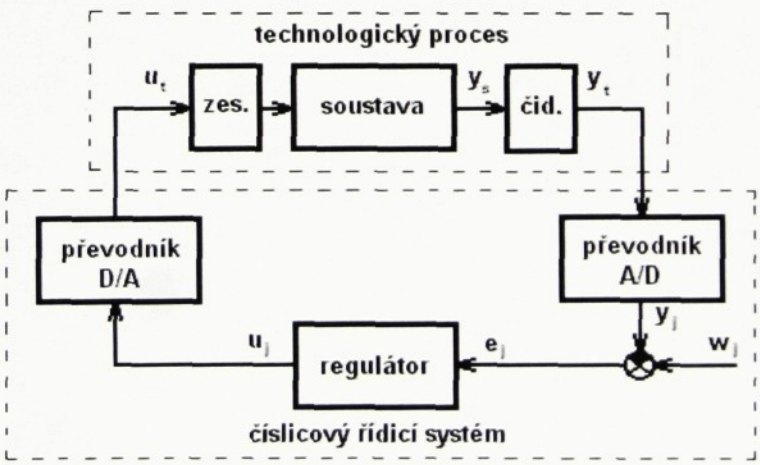
Při naprogramování ovládacího panelu s poměrně jednoduchým menu a displeji, zobrazující některé výstupy uživatelského programu, představovaly bohužel přeložené soubory 20 % celkové velikosti kódu programu pro PLC.

3. Regulovaná soustava

Pro účely zkoušení navrhovaného řídicího systému byla zvolena dynamická soustava nacházející se v laboratoři Katedry řídicí techniky. Jedná se o regulaci otáček tachodynama spojeného pružným hřídelem se stejnosměrným elektromotorem.

3.1. Číslicový regulační obvod

Blokové schéma zapojení pro regulaci soustavy s jedním vstupem a jedním výstupem číslicovým regulátorem je uvedeno na obr.3.1. K regulované soustavě je připojen pomocí AD a DA převodníků číslicový počítač. Obecně platí, že počítač i oba převodníky jsou ve skutečnosti souhrnně reprezentovány tzv. *číslicovým řídicím* systémem. Jedná se o spojitý technologický proces s jednou akční veličinou u_t a jednou fyzikální regulovanou veličinou y_s.Vstupní veličinou číslicového řídicího systému je regulovaná veličina y_t a řídicí veličina w_j, která je posloupností žádaných hodnot v diskrétních časových okamžicích *j, j+1, j+2....* Výstupní veličinou řídicího systému je akční veličina u_t.



Obr.3.1 Blokové schéma zapojení pro regulaci číslicovým regulátorem

Na obr.3.1 jsou jednotlivé veličiny:

u_t ... akční veličina regulované soustavy

y_t ... výstupní veličina regulované soustavy (regulovaná veličina)

y_s … výstupní fyzikální veličina regulované soustavy (fyz. regulovaná veličina)

w_j … žádaná hodnota výstupní veličiny, posloupnost hodnot (řídící veličina)

y_j … měřený výstup soustavy v diskrétním časovém okamžiku j

u_j … akční veličina v diskrétním časovém okamžiku j

e_j … regulační odchylka v diskrétním časovém okamžiku j

3.1.1. Z přenos lineární dynamické soustavy

Pro účely řízení je nutné znát diskrétní přenos regulované soustavy, který v číslicovém regulačním obvodu popisuje chování spojité dynamické soustavy.

Ryze dynamické soustavy řádu r bez dopravního zpoždění mají diskrétní přenos ve tvaru

$$\frac{y}{u} = z^{-1}\frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_{r-1} z^{-(r-1)}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_r z^{-r}} = z^{-1}\frac{B}{A}, \tag{3-1}$$

A je polynom stupně r,

B je polynom stupně r-1.

Diferenční rovnice modelu soustavy je ve tvaru

$$y_j + a_1 y_{j-1} + a_2 y_{j-2} + \ldots + a_r y_{j-r} = b_0 u_{j-1} + b_1 u_{j-2} + b_2 u_{j-3} + \ldots + b_{r-1} u_{j-r}. \tag{3-2}$$

3.1.2. Z přenos číslicového regulátoru

Číslicový regulátor má diskrétní přenos v tomto tvaru

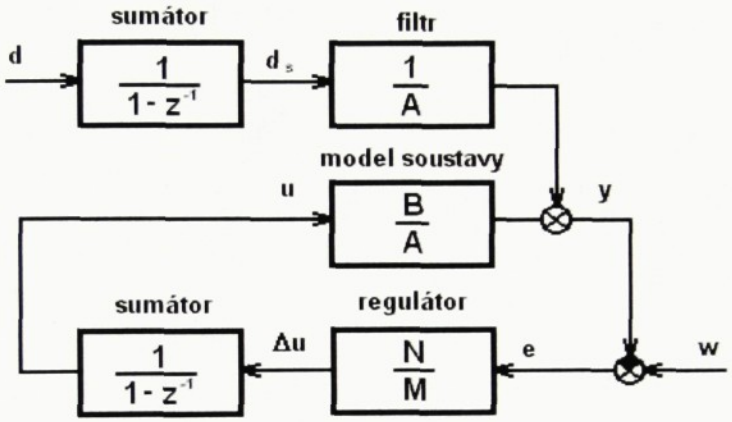
$$\frac{\Delta u}{-y} = \frac{n_0 + n_1 z^{-1} + n_2 z^{-2} + \ldots + n_r z^{-r}}{1 + m_1 z^{-1} + m_2 z^{-2} + \ldots + m_r z^{-r}} = \frac{N}{M}, \tag{3-3}$$

M, N jsou polynomy stupně r.

Diferenční rovnice číslicového regulátoru je pak

$$\Delta u_j + m_1 \Delta u_{j-1} + m_2 \Delta u_{j-2} + \ldots + m_r \Delta u_{j-r} = -n_0 y_j - n_1 y_{j-1} - n_2 y_{j-2} - \ldots - n_r y_{j-r} . \qquad (3\text{-}4)$$

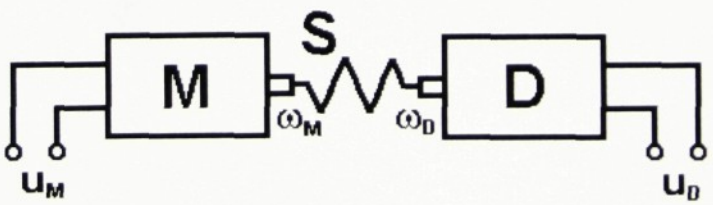
Struktura číslicového regulačního obvodu je na obr.3.2. Podle uvedeného přenosu filtru se jedná o tzv. regresní model regulované soustavy.



Obr.3.2 Číslicový regulační obvod

3.2. Popis a zapojení soustavy

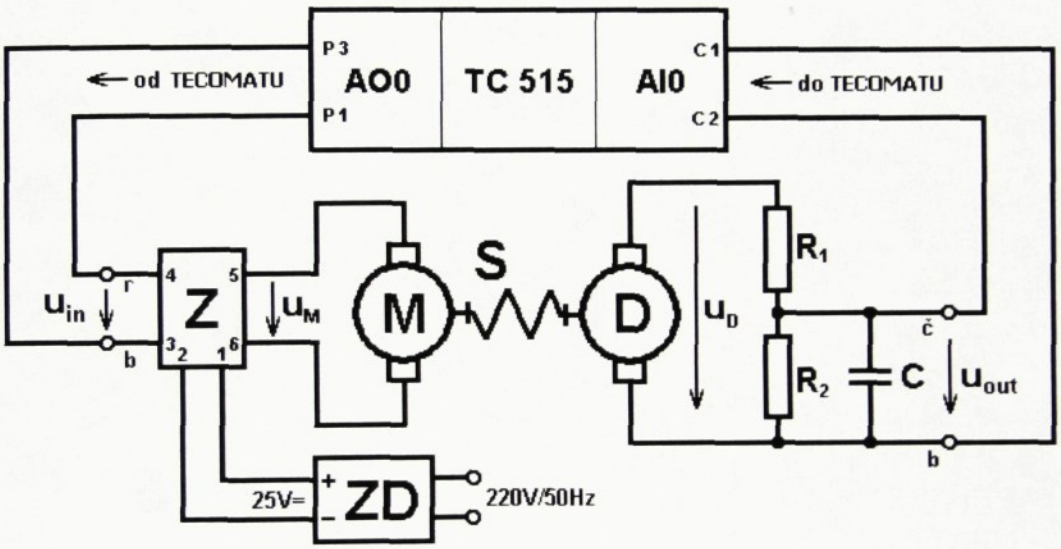
Dynamickou soustavu na obr.3.3 tvoří stejnosměrný elektromotor *M*, který je pomocí pružné spojky *S* spojen se stejnosměrným tachodynamem *D*. Soustava je buzena napětím u_M, výstupní měřenou veličinou je napětí tachodynama u_D. Mechanickými fyzikálními veličinami jsou otáčky motoru ω_M a otáčky tachodynama ω_D.



Obr.3.3 Dynamická soustava

Tato soustava byla připojena k měřící kartě PCL812-PG nacházející se v příslušném PC, kde je používána v prostředí Matlab, Simulink s využitím RealTime Toolboxu.

Z důvodu zachování současné laboratorní úlohy jsem provedl minimální úpravy v celém zapojení. Na výstup soustavy, tedy na svorky výstupního děliče, jsem připojil svorky analogového vstupu *AI0* vstupní/výstupní jednotky PLC TC515. A na vstupní svorky výkonového zesilovače *Z* jsem zařadil přepínač mající funkci *„ŘÍZENÍ PC/ŘÍZENÍ TECOMAT"*. Tedy svorky analogového výstupu *AO0* vstupní/výstupní jednotky PLC TC515 jsou připojeny k výkonovému zesilovači jen v režimu *„ŘÍZENÍ TECOMAT"*. Celkové schéma zapojení je zobrazeno na obr.3.4.



Obr.3.4 Celkové zapojení regulované soustavy a řídicího systému TECOMAT

Jednotlivé části mají následující význam:

M ... motor s permanentním buzením, P2TV369 MEZ Náchod, 24V, 2000 ot/min

D ...dynamo (motor v režimu dynama),P2TV369 MEZ Náchod, 24V, 2000 ot/min

S ... pružná spojka

Z ... zesilovací a výkonový člen

ZD ...napájecí zdroj, DBP 271, ZPA Děčín, 25V/10A

R_1 ... odpor děliče napětí, 33kΩ

R_2 ... odpor děliče napětí, 8kΩ

C ... filtrační kondenzátor

AO0 ...analogový výstup TC515

AI0 ... analogový vstup TC515

Popis signálů:

u_{in} ... řídící napětí z TC515, rozsah 0-5V

u_M ... napájecí napětí motoru, rozsah 0-24V

u_D ... napětí na tachodynamu, rozsah 0-24V

u_{out} ... vstupní napětí do TC515, rozsah 0-5V

3.3. Rozsahy vstupů a výstupů

Vzhledem k tomu, že analogové vstupy a výstupy TC515 mají rozsah 0 až 10V, ale napěťové úrovně v soustavě jsou 0 až 5V, bylo nutné používat vždy jen polovinu z celkového rozsahu obou převodníků. Tedy binární reprezentace vstupu je 0 až 2047, pro 12-ti bitový AD převodník a binární reprezentace výstupu je 0 až 127, pro 8-mi bitový DA převodník.

Otáčky motoru ani tachodynama nelze bohužel měřit přímo, např. stroboskopem, ale pouze vypočítat z odpovídajícího napětí na příslušných svorkách, podle rovnice (3-5). Konkrétně lze vycházet ze štítkových údajů tachodynama, tedy uvažovat, že napětí 24V odpovídají otáčky 2000ot/min.

$$n = \frac{2000}{24} u_D [ot./min] \tag{3-5}$$

Dále je možné zjistit hodnoty skutečného napětí na výstupním a vstupním převodníku a používat je pro vnitřní reprezentaci akční a výstupní veličiny v programu. Takto přepočtené hodnoty napětí z binárních reprezentací DA a AD převodníků lze získat ze vztahů (3-6) a (3-7).

$$u_V = \frac{9,96}{256} u_B [V] \tag{3-6}$$

$$y_V = \frac{9,96}{4096} y_B [V] \tag{3-7}$$

Když jsem se rozhodoval jakým způsobem budu reprezentovat hodnoty akční a výstupní veličiny, volil jsem nejjednodušší variantu, a to používat hodnoty přímých binárních reprezentací vstupního a výstupního převodníku, bez dalších přepočtů na napětí či otáčky. Důvodem bylo především to, že systémový analyzátor pracuje jen s celými čísly, tedy proměnnými typu word.

Všechny další hodnoty a průběhy akční a výstupní veličiny uvedené v této práci, budou tedy dále udávány bezrozměrně. Jejich označení pro akční veličinu je u, pro výstupní veličinu y.

3.4. Měření statické charakteristiky

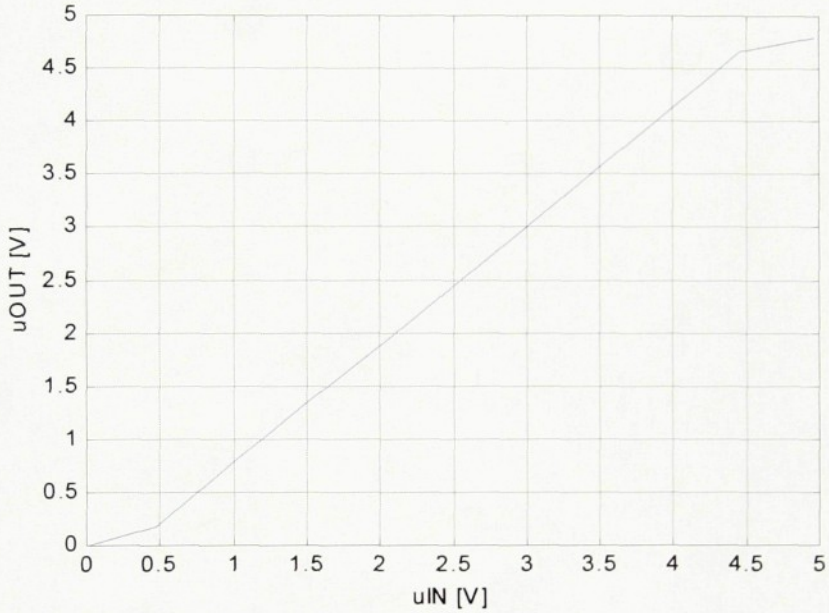
Měření statické charakteristiky regulované soustavy a měření převodních charakteristik jsem provedl pomocí programu Static.jlr. Převodní charakteristiky jsou závislosti mezi hodnotami vstupních a výstupních proměnných v prostředí Epos a konkrétních fyzikálních veličin, tj. napětích měřených přímo na svorkách motoru a tachodynama. Naměřené hodnoty jsou v tab.3.1, označení veličin odpovídá obr.3.4.

Hodnoty označené u_B a y_B odpovídají přímým binárním reprezentacím analogového výstupu a vstupu. Přísluší jim konkrétní proměnné programu *VYSTUP* a *VSTUP* v prostředí Epos. Dále je tabulka doplněna o hodnoty označené u_V a y_V, představující přepočítané binární reprezentace analogového výstupu a vstupu na hodnoty napětí. Těm přísluší proměnné *UV* a *YV*.

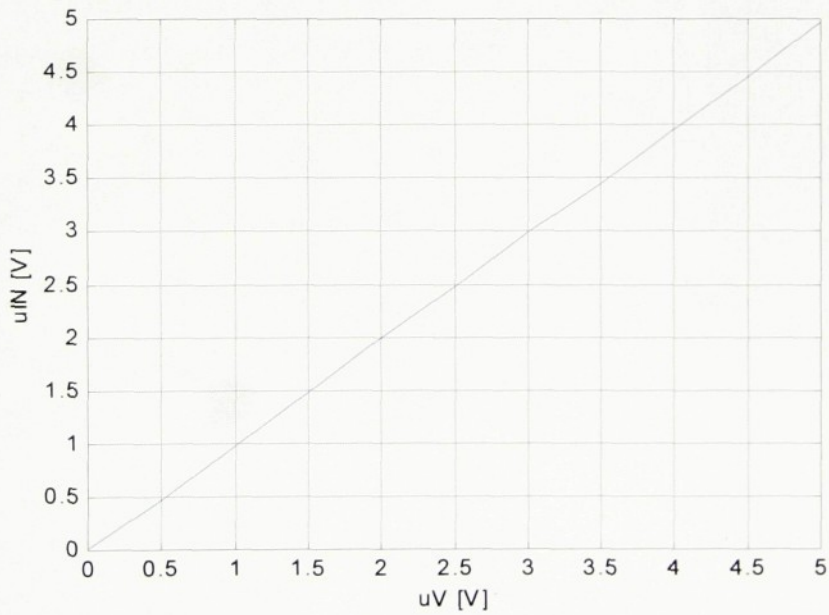
Tab. 3.1 Naměřené hodnoty statických a převodních charakteristik

u_V[V]	u_B[-]	u_{in}[V]	u_M[V]	u_D[V]	u_{out}[V]	y_B[-]	y_V[V]
0,00	0	0,00	0,18	0,00	0,00	0	0,00
0,50	12	0,48	2,12	0,75	0,18	60	0,15
1,00	25	0,98	4,59	3,32	0,76	293	0,71
1,50	38	1,48	7,06	5,86	1,33	525	1,27
2,00	51	1,99	9,53	8,41	1,86	756	1,84
2,50	64	2,48	12,01	10,97	2,42	992	2,41
3,00	77	2,99	14,49	13,52	2,99	1234	2,98
3,50	89	3,45	16,77	15,91	3,52	1444	3,50
4,00	102	3,96	19,25	18,47	4,09	1680	4,09
4,50	115	4,46	21,7	21,00	4,67	1916	4,66
5,00	127	4,97	22,30	21,60	4,79	1971	4,80

Naměřená statická charakteristika je sestrojená do grafu na obr 3.5 a převodní charakteristika je na obr.3.6.



Obr.3.5 Statická charakteristika dynamického systému



Obr.3.6 Převodní charakteristika

Ze statické charakteristiky regulované soustavy na obr.3.5 je zřejmé, že dynamická soustava se dá považovat za lineární zhruba pro hodnoty napětí u_{IN} od 0,5V do 4,5V.

4. Teoretická část

K vytvoření požadovaného adaptivního regulátoru se bylo třeba seznámit s odpovídajícími matematickými metodami a algoritmy. Pro identifikaci lineární dynamické soustavy je to metoda LD rozkladu nazývaná také metodou nejmenších čtverců s exponenciálním zapomínáním a pro syntézu to byla metoda dynamického programování pro návrh číslicového regulátoru. Prostředkem k seznámení se s touto problematikou, mi byla v textu dále uvedená literatura.

4.1. Identifikace lineární dynamické soustavy metodou LD rozkladu

Úkolem průběžné identifikace je provést na základě pozorovaných - měřených vstupů a výstupů až do časového okamžiku *k* odhad parametrů prediktoru podle zvoleného kritéria. Odhady koeficientů prediktoru budou dále využity k návrhu algoritmu řízení - syntézy.

Identifikačním algoritmem průběžného odhadování parametrů metodou LD rozkladu je rekurzivní metoda nejmenších čtverců s exponenciálním zapomínáním. Algoritmus této metody je podrobně vysvětlen v [5], odkud jsem také čerpal tyto podklady.

Kritérium výběru vektoru parametrů bude

$$J_k(P) = \sum_{j=0}^{k-1} \varepsilon_j^2 \rightarrow min, \tag{4-1}$$

kde

$$\varepsilon_j = y_j - y_j^P = x^T(j)\begin{bmatrix} I \\ P \end{bmatrix}, \tag{4-2}$$

y_j...je měřený výstup systému v okamžiku j,

y^P_j...je výstup prediktoru v okamžiku j,

P...je odhad parametrů prediktoru.

Vzhledem k tomu, že bodové odhady jsou určeny pro adaptivní řízení, je třeba klást na identifikaci metodou nejmenších čtverců další podmínky a požadavky.

Realizovat zapomínání, tj. zajistit, aby nejstarším měřeným hodnotám byla přisuzována co nejmenší váha tak, aby se co nejvíce uplatňovala nejnovější měření. Zajistit průběžné upřesňování odhadů na základě nového měření. Realizovat numericky jednoduchý algoritmus, který zaručuje rychlou konvergenci odhadů s malými nároky na operační paměť a výpočetní čas.

Zapomínání je vyjádřeno mocninou faktoru exponenciálního zapomínání ve tvaru

$$\bar{\varepsilon}_{k-j} = \varphi^j \varepsilon_{k-j}. \tag{4-3}$$

Modifikované kritérium respektující exponenciální zapomínání je

$$J_k(P) = \sum_{j=0}^{k-1} \left(\varphi^j \varepsilon_{k-j}\right)^2. \tag{4-4}$$

Dosazením z (4-2) do (4-4) dostaneme

$$J_k(P) = \left(1; P^T\right) \left[\sum_{j=0}^{k-1} \left(x_{k-j} x_{k-j1}^T \varphi^{2j}\right)\right] \begin{bmatrix} 1 \\ P \end{bmatrix}.$$

Hodnotu kvadratického kritéria je možno vyjádřit jako kvadratickou formu

$$J_k(P) = \left(1; P^T\right) V_k \begin{bmatrix} 1 \\ P \end{bmatrix}, \tag{4-5}$$

kde matice

$$V_k = \sum_{j=0}^{k-1} \left(x_{k-j} x_{k-j1}^T \varphi^{2j}\right).$$

Aby bylo možné splnit požadavek průběžného odhadování parametrů, je nutno aktualizaci matice V_k provádět rekurzivním výpočtem

$$V_k = x_k x_k^T + \varphi^2 V_{k-1}. \tag{4-6}$$

Pro počáteční hodnotu matice $V_k=0$ takto definovaná matice zajišťuje, že kvadratická forma (4-5) je vždy nezáporná-positivně definitní. Tím současně zajišťuje existenci konečného vektoru parametrů P. Je nutné zajistit positivní semidefinitnost matice V_k, v opačném případě bude mít vektor P neomezené složky. Tento problém se řeší pomocí vhodného rozkladu matice, např. numericky úsporného algoritmu, který

využívá alternativního LD-rozkladu. Za předpokladu, že matice V_k je regulární, positivně definitní, je možno provést následující rozklad

$$V_k^{-1} = M = L_k D_k L_k^T, \tag{4-7}$$

kde

D_k...je diagonální matice s kladnými prvky,

L_k...je dolní trojúhelníková matice s jednotkovou hlavní diagonálou.

Matice D_k, L_k lze rozdělit tak, aby byl vyčleněn predikovaný výstup y_k z vektoru x_k.

$$x_k = \begin{bmatrix} y_k \\ z_k \end{bmatrix}, \qquad D = D_k = \begin{bmatrix} D_y & 0 \\ 0 & D_z \end{bmatrix}, \qquad L = L_k = \begin{bmatrix} 1 & 0 \\ L_{zy} & L_z \end{bmatrix}.$$

Matice V_k se dá vyjádřit ve tvaru

$$V_k = \left(L_k D_k L_k^T\right)^{-1} = \left(L_k^{-1}\right)^T D_k^{-1} L_k^{-1},$$

kritérium má pak tvar

$$J_k(P) = \left(1; P^T\right)\left(L_k^{-1}\right)^T D_k^{-1} L_k^{-1} \begin{bmatrix} 1 \\ P \end{bmatrix}. \tag{4-8}$$

Dalším krokem je analytické určení inverze trojúhelníkové matice L_k, a dále dosazením do (4-8) je kvadratické kritérium ve tvaru

$$J_k(P) = \left(1; P^T\right) \begin{bmatrix} 1 & -L_z^{-1} L_{zy} \\ 0 & L_z^{-1} \end{bmatrix} \begin{bmatrix} D_y^{-1} & 0 \\ 0 & D_z^{-1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -L_z^{-1} L_{zy} & L_z^{-1} \end{bmatrix} \begin{bmatrix} 1 \\ P \end{bmatrix}.$$

Roznásobením a převedením na tzv. úplný čtverec je kvadratické kritérium v tomto tvaru

$$J_k(P) = D_y^{-1} + \left(P - L_{zy}\right)^T \left(L_z\right)^{-1} D_k^{-1} \left(L_z\right)^{-1} \left(P - L_{zy}\right). \tag{4-9}$$

Minimum funkcionálu $J_K(P)$ ve výrazu (4-9) je zřejmě

$$\min_P J = D_y^{-1}, \quad \text{pro vektor parametrů P = Lzy.} \tag{4-10}$$

Postup, který je nutno provést, aby bylo možné určit ze vzorce (4-10) vektor hledaných parametrů je následující:

a) v daném časovém okamžiku k je třeba provést aktualizaci matice V_K dle vztahu (4-6)

b) rozklad matice V_k^{-1} dle algoritmu alternativního LD rozkladu

c) parametry jsou pak dány rovností (4-10).

Aby nebylo nutné provádět v každém identifikačním kroku celou faktorizaci matice V_{k-1}, je potřeba najít rekurzivní algoritmus, který vyčísluje přímo matice L_k a D_k, zároveň tak splnit požadavek na identifikaci, aby bylo dosaženo numericky jednoduchého algoritmu.

Faktorizace matice V_k^{-1} byla zavedena rovností (4-7), s využitím vztahu (4-6) musí platit

$$L_k D_k L_k^T = V_k^{-1} = \left(\varphi^2 V_{k-1} + x_k x_k^T\right)^{-1}.$$

Pokud je nejdříve vyjádřena matice V_k^{-1} ve tvaru

$$V_{k-1} = \left(L_{k-1} D_{k-1} L_{k-1}^T\right)^{-1}. \tag{4-12}$$

Pak je pravou stranu rovnosti (4-11) s využitím vlastností inverzních matic a rovnosti (4-12) možno upravit do tvaru

$$L_k D_k L_k^T = \left[\varphi^2 \left(L_{k-1} D_{k-1} L_{k-1}^T\right)^{-1} + x_k x_k^T\right]^{-1}$$

$$L_k D_k L_k^T = \frac{1}{\varphi^2} L_{k-1} M L_{k-1}^T, \tag{4-13}$$

kde

$$M = \left(D_{k-1}^{-1} L_{k-1}^T \frac{x_k x_k^T}{\varphi^2} L_{k-1}^T \right)^{-1} = \left(D_{k-1}^{-1} f \frac{1}{\varphi^2} f^T \right)^{-1}, \tag{4-14}$$

$$f = L_{k-1}^T x_k. \tag{4-15}$$

Aktualizace matice V_k^{-1} závisí tedy na předchozí hodnotě matice L_{k-1} a na matici M. Matice M je pozitivně definitní a existuje její rozklad do tvaru

$$M = H \, \overline{D} \, H^T \tag{4-16}$$

Rovnost (4-13) je pak možno zapsat do tvaru

$$L_k D_k L_k^T = \frac{1}{\varphi^2} L_{k-1} H \, \overline{D} \, H^T L_{k-1}^T \tag{4-17}$$

Srovnáním levé a pravé strany rovnosti (4-17) lze dostat hledané rekurzivní vztahy pro aktualizaci matic L_k, D_k a platí

$$L_k = L_{k-1} H\,, \tag{4-18}$$

$$D_k = \frac{\overline{D}}{\varphi^2}\,. \tag{4-19}$$

Aby bylo možné provádět faktorizaci, musí být známy koeficienty matice *M*.

$$M = \left(D_{k-1}^{-1} + f \frac{1}{\varphi^2} f^T \right)^{-1} = D_{k-1} - D_{k-1} \frac{f\, f^T}{\varphi^2 + f^T D_{k-1} f} D_{k-1} \tag{4-20}$$

Pro přehlednost po vypuštění časového indexu *k-1* je možno prvky M_{ij} matice M vyjádřit pomocí vzorců

$$M_{ii} = D_{ii} - \frac{D_{ii}^2 f_i^2}{\varphi^2 + \sum_{i=1}^{n} f_i^2 D_{ii}} \tag{4-21}$$

$$M_{ii} = -\frac{D_{ii} f_i \; D_{jj} f_j}{\varphi^2 + \sum_{i=1}^{n} f_i^2 D_{ii}} \tag{4-22}$$

Z analytického vyjádření faktorizace pak pro rozklad (4-16) plynou vzorce

$$\overline{D}_{ii} = M_{ii} - \sum_{l=1}^{i-1} \overline{D}_{ll} H_{il}^2 \qquad \text{pro } i \geq 1\,, \tag{4-23}$$

$$H_{ji} = \frac{1}{\overline{D}_{ii}} \left(M_{ji} - \sum_{l=1}^{i-1} \overline{D}_{ll} H_{il} H_{jl} \right) \qquad \text{pro } j > i\,. \tag{4-24}$$

Identifikační algoritmus metodou LD rozkladu je naprogramován v Matlabu, soubor LdfRun.m je výpočetním jádrem obslužného programu Ld2Ident.m. Příloha č.1 obsahuje výpis zdrojového programu, oba programy jsou dostupné v elektronické podobě v příloze č.4.

Vstupní parametry jsou *FR* - koeficient zapomínání, *NR* - řád soustavy, *DA* - vektor vstupních dat *(y_k y_{k-1} ... , u_{k-1} ...,1)* dle struktury matematického modelu. Výstupním parametrem je *VU* - vektor hledaných parametrů *(a_1 a_2 ..., b_0 b_1...)*. V každém identifikačním kroku je nutno zajistit aktualizaci vektoru dat *DA*. Prvek *DA(1)* je měřená hodnota y_k , další prvky je nutno obsadit minulými hodnotami výstupu a vstupu y_{k-i} , u_{k-j} dle struktury matematického modelu. Tuto manipulaci s naměřenými daty a cyklické zpracování programu zajišťuje obslužný program.

4.2. Metoda dynamického programování pro návrh číslicového regulátoru

Obecnou úlohou dynamického programování může být sekvenční rozhodovací úloha s konečným počtem stavů, kdy se hledá cesta orientovaným grafem tak, aby součet penalizací po této cestě byl minimální. Pro číslicové řízení lineárních dynamických systémů je tato úloha zevšeobecněna. Zde je tato metoda použita ve smyslu minimalizace rozšířeného kvadratického kritéria kvality regulace.

Metoda dynamického programování používá stavový popis regulované soustavy a výsledkem této metody je optimální stavový regulátor. Pro připojení stavového regulátoru do regulačního obvodu se nepoužívá klasický zpětnovazební regulátor, ale je odvozena metoda pro výpočet diferenční rovnice regulátoru z maticové rovnice stavového regulátoru a popisu regulované soustavy. Tímto postupem, při použití rozšířeného kvadratického kritéria je získána diferenční rovnice regulátoru stejná jako algebraickou metodou, samozřejmě při minimalizaci stejného kritéria.

Vzhledem k rozsahu této práce není možné na tomto místě provést odpovídající rozbor této metody. Proto se odvolávám na literaturu [4], která mi byla podkladem ke studiu, kde je vše potřebné důkladně zpracováno.

Jedná se především o princip dynamického programování použitého pro číslicové řízení lineárních dynamických systémů. Dále je zde uveden výpočet stavového regulátoru metodou dynamického programování pro soustavu s jedním vstupem a jedním výstupem a odvození výpočtu diferenční rovnice číslicového regulátoru z rovnice stavového regulátoru a popisu regulované soustavy.

Pro algoritmické zjednodušení rekurzivního výpočtu stavového regulátoru, pro soustavu s jedním vstupem a jedním výstupem, je uvažováno rozšířené kvadratické kritérium ve tvaru

$$J = \sum_{j=0}^{n-1} \left(y_j^2 + \kappa \Delta u_j^2 \right) + y_n^2 \qquad (4\text{-}25)$$

V dynamickém programování rekurzivní výpočtové kroky opakujeme a pro řízení uvažujeme limitní matice, které konvergují k ustálenému řešení. Z toho důvodu lze uvedené rozšířené kvadratické kritérium zapsat ve tvaru

$$J = \lim_{n\to\infty} \sum_{j=0}^{n} \left(y_j^2 + \kappa \Delta u_j^2 \right) = \sum_{j=0}^{\infty} \left(y_j^2 + \kappa \Delta u_j^2 \right) \qquad (4\text{-}26)$$

kde κ je váhový koeficient, jehož zvětšováním se regulační pochody zpomalují.

Matice stavového regulátoru v případě soustavy s jedním vstupem a jedním výstupem je řádkovým vektorem. Při označení matice optimálního stavového regulátoru R_j, pak posloupnost těchto matic konverguje k matici označené R_0. V kontrolním příkladě naprogramovaného algoritmu se jednalo řádově o desítky rekurzivních optimalizačních kroků výpočtu, kdy je dosaženo prakticky ustálené hodnoty matice R_0.

Algoritmus metody dynamického programování pro návrh parametrů číslicového regulátoru ze známého diskrétního modelu soustavy je naprogramován v Matlabu, soubor DynRun.m je výpočetním jádrem obslužného programu Dyn2Reg.m. Příloha č.2 obsahuje výpis zdrojového programu, oba programy jsou dostupné v elektronické podobě v příloze č.4.

Vstupními parametry jsou *NR* - řád regulované soustavy, *Kapa* - koeficient tlumení regulace a diskrétní přenos regulované soustavy ve tvaru: *PolA* - jmenovatel přenosu s koeficienty *(1 a_1 a_2...)*, *PolB* - čitatel s koeficienty *(b_0 b_1 ...)*. Výstupními parametry jsou *StateReg* - vektor stavového regulátoru a diskrétní přenos navrženého regulátoru ve tvaru: *PolM* - jmenovatel přenosu regulátoru *(1 m_1 m_2...)*, a *PolN* - čitatel přenosu regulátoru *(n_0 n_1 n_2...)*. Obslužný program zajišťuje inicializaci proměnných a opakování rekurzivních výpočtových kroků.

5. Realizační část

V této kapitole bych rád popsal, jakým způsobem jsem postupoval na praktické části mé práce, tedy naprogramování zvolených algoritmů a jejich odzkoušení na příkladech. Dále pak vytvoření programu v jazyce JLR pro systém TECOMAT, a nakonec ověření funkce naprogramovaného adaptivního regulátoru na regulované soustavě.

Vzhledem k potřebě průběžně kontrolovat správnost dílčích výsledků, jsem nejdříve přistoupil k naprogramování a odladění algoritmů identifikace a syntézy v prostředí MATLAB, kde jsem výsledky zkontroloval a porovnal s dostupnými funkcemi, a poté teprve implementoval v prostředí Epos na řídicí systém TECOMAT.

5.1. Identifikace - algoritmus průběžného odhadování parametrů metodou LD rozkladu

V následujících kapitolách jsou uvedeny kontrolní příklady pro ověření správnosti naprogramovaného algoritmu. Všechny uvedené programy jsou v příloze P.4. Jedná se o porovnání metody LD rozkladu s funkcí identifikačního toolboxu arx a srovnání výsledků metody LD rozkladu naprogramované v syntaxi jazyka JLR, při off-line běhu programu, s předchozími výsledky.

Všechny tři programy pracují se stejným souborem naměřených dat regulované soustavy.

5.1.1. Algoritmus LD rozkladu naprogramovaný v prostředí Matlab

Kontrolní příklad, program Ld2Ident.m je výpočet koeficientů přenosu diskrétního modelu a simulace jeho odezvy.

Diskrétní model dynamické soustavy je ve tvaru:

$$(1+a_1z^{-1} + a_2z^{-2} + a_3z^{-3} + a_4z^{-4})\, y = z^{-1}(b_0 + b_1z^{-1} + b_2z^{-2} + b_3z^{-3})\, u$$

$$y_j + a_1 y_{j-1} + a_2 y_{j-2} + a_3 y_{j-3} + a_4 y_{j-4} = b_0 u_{j-1} + b_1 u_{j-2} + b_2 u_{j-3} + b_3 u_{j-4} \qquad (5\text{-}1)$$

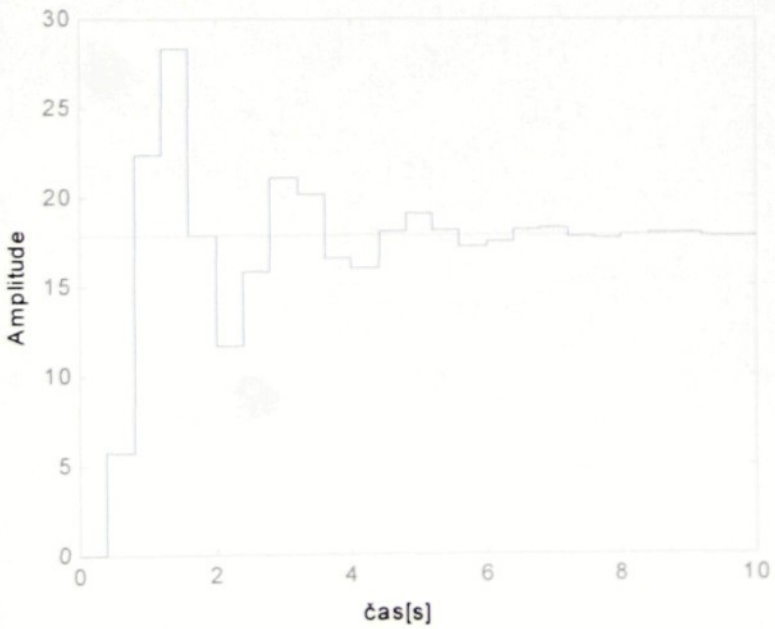
Diskrétní obrazový přenos:

$$F(z) = \frac{z^{-1}\left(b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}\right)}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + a_4 z^{-4}} \qquad (5\text{-}2)$$

Výsledný obrazový přenos vypočtený programem Ld2Ident.m:

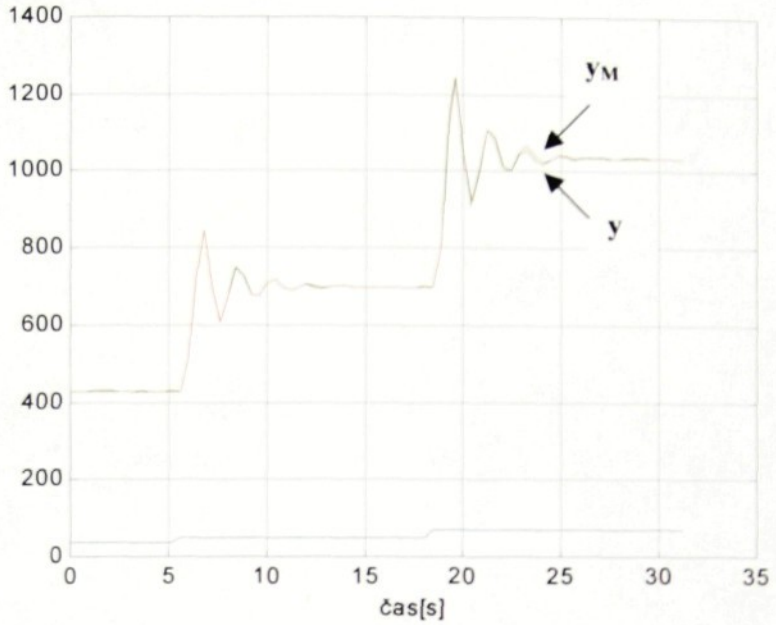
$$FM(z) = \frac{z^{-1}\left(5{,}6999 + 14{,}1847 z^{-1} + 0{,}7404 z^{-2} - 6{,}5138 z^{-3}\right)}{1 - 0{,}4456 z^{-1} + 0{,}3954 z^{-2} + 0{,}0059 z^{-3} - 0{,}1662 z^{-4}} \qquad (5\text{-}3)$$

Přechodová charakteristika vypočteného diskrétního modelu, vykreslená pomocí funkce step je na obr.5.1.



Obr.5.1 Přechodová charakteristika diskrétního modelu FM

Odezva diskrétního modelu vypočtená programem Ld2Ident.m je vykreslená pomocí funkce plot na obr.5.2.



Obr.5.2 Odezva diskrétního modelu FM: naměřené hodnoty výstupní veličiny y, odezva identifikovaného diskrétního modelu regulované soustavy y_M

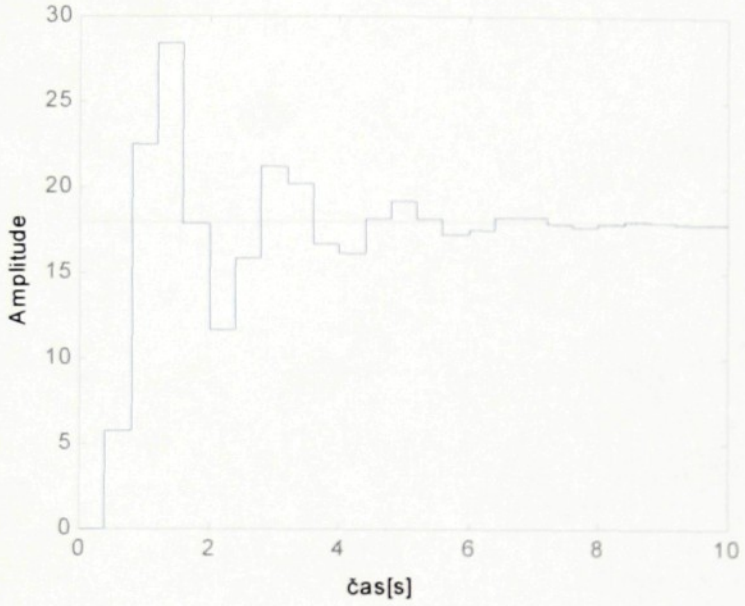
5.1.2. Funkce ARX identifikačního toolboxu v prostředí Matlab

Kontrolní příklad, program Arx2Ident.m je výpočet koeficientů přenosu diskrétního modelu pomocí funkce arx a simulace jeho odezvy pomocí funkce idsim.

Výsledný obrazový přenos vypočtený programem Arx2Ident.m:

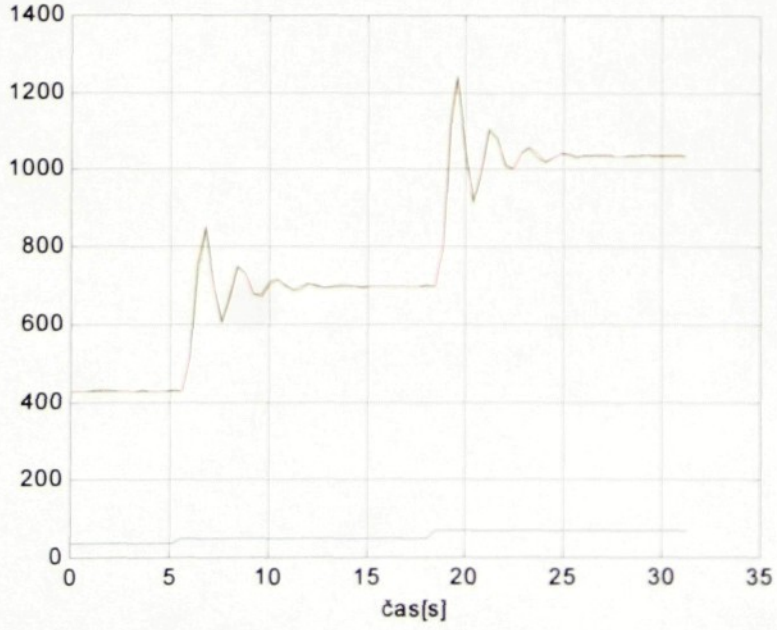
$$FA(z)=\frac{z^{-1}\left(5{,}7412+14{,}1227z^{-1}+0{,}5429z^{-2}-6{,}6544z^{-3}\right)}{1-0{,}4565z^{-1}+0{,}3929z^{-2}+0{,}0010z^{-3}-0{,}1699z^{-4}} \qquad (5\text{-}4)$$

Přechodová charakteristika vypočteného diskrétního modelu vykreslená pomocí funkce step je na obr.5.3.



Obr.5.3 Přechodová charakteristika diskrétního modelu FA

Odezva diskrétního modelu vypočtená funkcí idsim, vykreslená pomocí funkce plot je na obr.5.4.



Obr.5.4 Odezva diskrétního modelu FA: naměřené hodnoty výstupní veličiny y, odezva identifikovaného diskrétního modelu regulované soustavy y_A

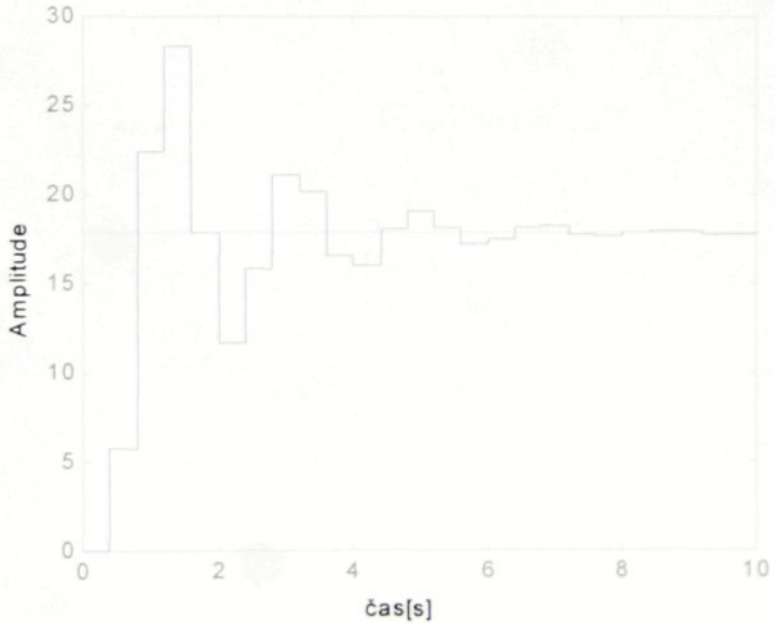
5.1.3. Algoritmus LD rozkladu naprogramovaný v prostředí Epos

Kontrolní příklad, program Ident2OffLine.jlr je výpočet koeficientů přenosu diskrétního modelu a simulace jeho odezvy při off-line běhu programu na PLC TC515.

Výsledný obrazový přenos vypočtený programem Ident2OffLine.jlr:

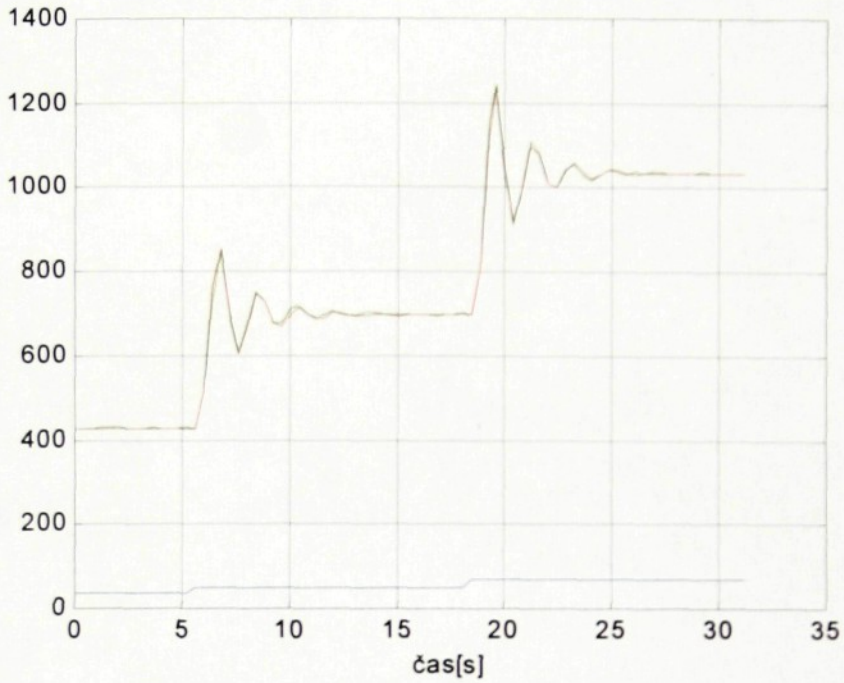
$$FT(z)=\frac{z^{-1}\left(5{,}6999+14{,}1840z^{-1}+0{,}7382z^{-2}-6{,}5155z^{-3}\right)}{1-0{,}4457z^{-1}+0{,}3954z^{-2}+0{,}0059z^{-3}-0{,}1663z^{-4}} \qquad (5\text{-}5)$$

Přechodová charakteristika vypočteného diskrétního modelu vykreslená pomocí funkce step je na obr.5.5.

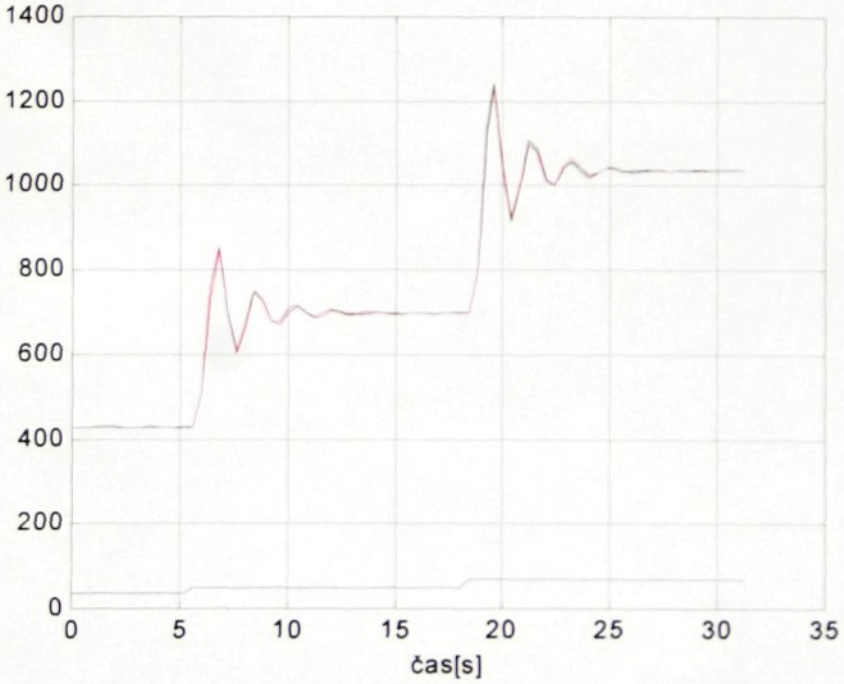


Obr.5.5 Přechodová charakteristika diskrétního modelu FT

Odezva diskrétního modelu vypočtená programem Ident2OffLine.jlr je vykreslená pomocí funkce plot na obr.5.6.



Obr.5.6 Odezva diskrétního modelu FT: měřené hodnoty výstupní veličiny y, odezva identifikovaného diskrétního modelu regulované soustavy y_T



Obr.5.7 Odezvy diskrétních modelů FM, FA, FT: naměřené hodnoty výstupní veličiny y, odezvy identifikovaných diskrétních modelů y_M ,y_A ,y_T

Pro porovnání uvádím na obr.5.7 jednotlivé výsledky zobrazené v jednom grafu. Všechny tři kontrolní programy dávají v podstatě shodné výsledky, u identifikace pomocí funkce arx se jednotlivé koeficienty přenosu diskrétního modelu mírně liší od výsledků, které dává metoda LD rozkladu.

5.2. Syntéza - metoda dynamického programování pro návrh adaptivního číslicového regulátoru

V dalších třech kapitolách jsou uvedeny kontrolní příklady pro ověření správnosti naprogramovaného algoritmu této metody. Jde o porovnání metody dynamického programování pro návrh adaptivního číslicového regulátoru ve smyslu minimalizace kvadratického kritéria s funkcí Control toolboxu lqry a srovnání výsledků této metody naprogramované v syntaxi jazyka JLR, při off-line běhu programu, s předchozími výsledky.

Pro všechny tři programy je zvolen stejný diskrétní model čtvrtého řádu dynamické soustavy.

5.2.1. Algoritmus metody dynamického programování v prostředí Matlab

Kontrolní příklad, program Dyn2Reg.m je výpočet koeficientů přenosu diskrétního regulátoru a simulace regulačního pochodu při poruše na výstupu modelu soustavy.

Diskrétní regulátor uvažujeme ve tvaru:

$$\Delta u(1+m_1 z^{-1} + m_2 z^{-2} + m_3 z^{-3} + m_4 z^{-4}) = -y(n_0 + n_1 z^{-1} + n_2 z^{-2} + n_3 z^{-3} + n_4 z^{-4})$$

$$\Delta u_j + m_1 \Delta u_{j-1} + m_2 \Delta u_{j-2} + m_3 \Delta u_{j-3} + m_4 \Delta u_{j-4} = -n_0 y_j - n_1 y_{j-1} - n_2 y_{j-2} - n_3 y_{j-3} - n_4 y_{j-4} \quad (5\text{-}6)$$

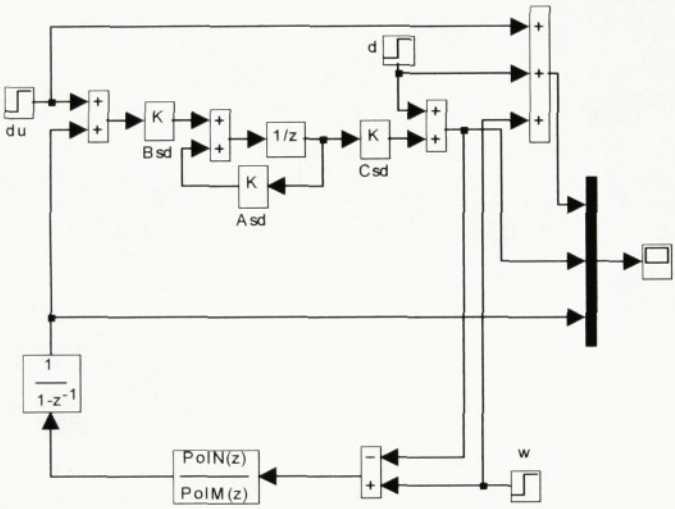
Přenos diskrétního regulátoru:

$$R(z)=\frac{n_0+n_1z^{-1}+n_2z^{-2}+n_3z^{-3}+n_4z^{-4}}{1+m_1z^{-1}+m_2z^{-2}+m_3z^{-3}+m_4z^{-4}} \tag{5-7}$$

Výsledný přenos diskrétního regulátoru vypočtený programem Dyn2Reg.m:

$$R(z)=\frac{10{,}1368-12{,}4251z^{-1}+5{,}3733z^{-2}-0{,}9512z^{-3}+0{,}0577z^{-4}}{1+0{,}5707z^{-1}+0{,}1673z^{-2}+0{,}0051z^{-3}+0{,}0000z^{-4}} \tag{5-8}$$

Simulační schéma regulačního obvodu je na obr.5.8.



Obr.5.8 Simulační schéma regulačního obvodu

Na obr.5.12. je zobrazen průběh regulačního pochodu při vstupu poruchy *d* do modelu soustavy. Zobrazené průběhy jsou: porucha *d*, výstupní veličina *y*, akční veličina *u*.

5.2.2. Funkce LQRY Control toolboxu v prostředí Matlab

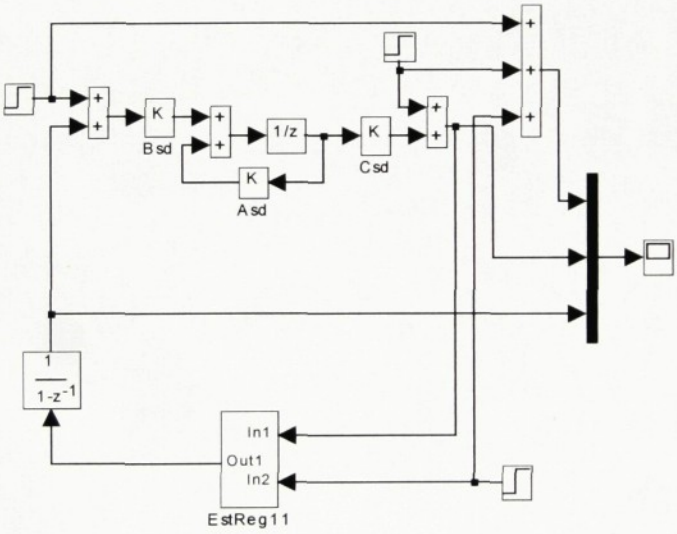
Kontrolní příklad, program EstArx12.m je výpočet optimálního zesílení matice *K* pomocí funkce lqry a odvození koeficientů přenosu diskrétního regulátoru pomocí speciálního estimátoru stavu. Dále pak simulace regulačního pochodu při poruše na výstupu modelu soustavy.

Výsledný přenos diskrétního regulátoru vypočtený programem EstArx12.m:

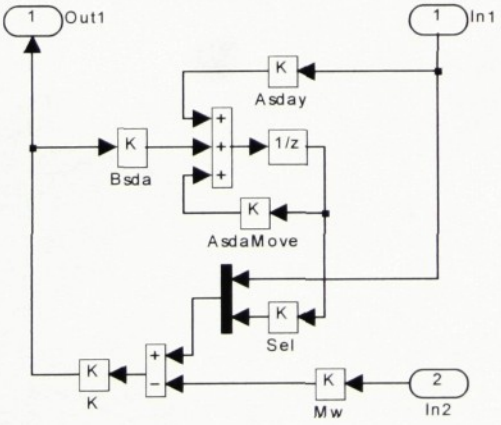
$$R(z)=\frac{10{,}1366-12{,}4253z^{-1}+5{,}3735z^{-2}-0{,}9512z^{-3}+0{,}0577z^{-4}}{1+0{,}5707z^{-1}+0{,}1673z^{-2}+0{,}0051z^{-3}+0{,}0000z^{-4}} \tag{5-9}$$

Simulační schéma regulačního obvodu se začleněným subsystémem estimátoru je na obr.5.9. Subsystém estimátoru je na obr.5.10. Schéma regulačního obvodu s odvozeným diskrétním regulátorem je na obr.5.11.

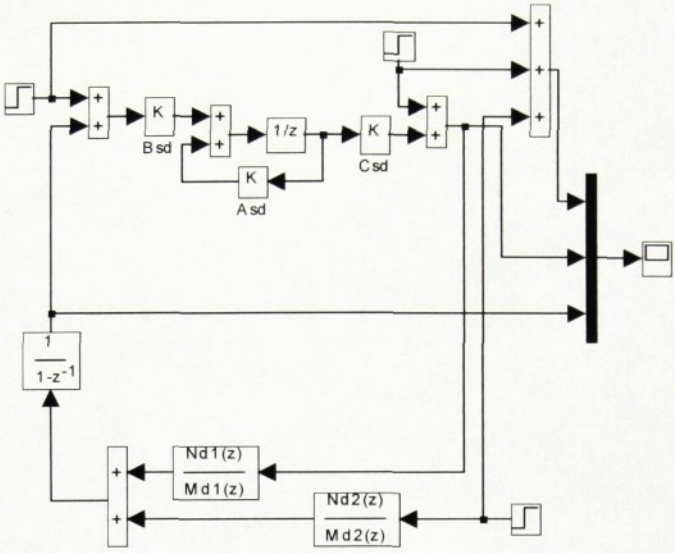
Na obr.5.12. je zobrazen průběh regulačního pochodu při vstupu poruchy *d* do modelu soustavy.



Obr.5.9 Simulační schéma regulačního obvodu se začleněným subsystémem estimátoru



Obr.5.10 Subsystém estimátoru



Obr.5.11 Schéma regulačního obvodu s odvozeným diskrétním regulátorem

5.2.3. Algoritmus metody dynamického programování v prostředí Epos

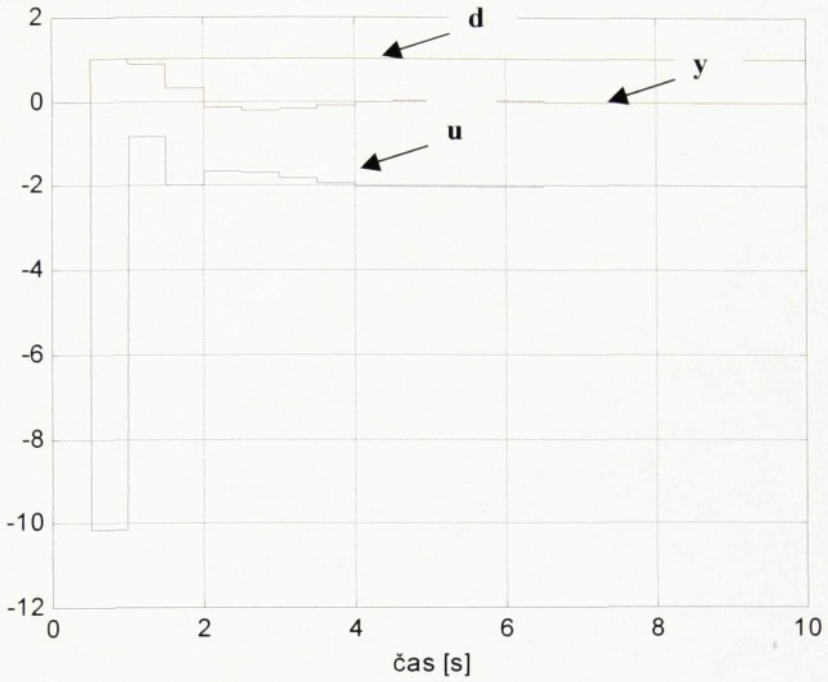
Kontrolní příklad, program Dyn2OffLine.jlr je výpočet koeficientů přenosu diskrétního regulátoru při off-line běhu programu na PLC TC515.

Výsledný přenos diskrétního regulátoru vypočtený programem Dyn2OffLine.jlr:

$$R(z)=\frac{10{,}1368-12{,}4251z^{-1}+5{,}3733z^{-2}-0{,}9512z^{-3}+0{,}0577z^{-4}}{1+0{,}5707z^{-1}+0{,}1673z^{-2}+0{,}0051z^{-3}+0{,}0000z^{-4}} \qquad (5\text{-}10)$$

Simulační schéma regulačního obvodu je stejné struktury jako schéma v příkladu Dyn2Reg.m, které je uvedeno na obr.5.8. Na obr.5.12 je zobrazen průběh regulačního pochodu při vstupu poruchy *d* do modelu soustavy.

Jelikož se jedná o rekurzivní algoritmus, výpočetní korky je nutné opakovat. Po deseti výpočetních krocích se jednotlivé koeficienty přenosu v podstatě již neměnily. Vypočtené koeficienty přenosu diskrétního regulátoru, ze všech tří kontrolních příkladů, si velmi přesně odpovídají. Tedy i příslušné regulační pochody jsou v podstatě stejné. Na obr.5.12 jsou zobrazeny všechny průběhy najednou.



Obr.5.12 Průběh regulačního pochodu při vstupu poruchy d do modelu soustavy: porucha d, výstupní veličina y, akční veličina u

5.3. Program pro řídicí systém TECOMAT

V následujících kapitolách je popsáno, jakým způsobem jsem postupoval při vytvoření programu adaptivního regulátoru v jazyce JLR pro systém TECOMAT.

Samotný úkol jsem si rozdělil do dvou hlavních částí. První je naprogramování algoritmu LD rozkladu a jeho odladění, jako samostatného programového bloku - identifikace. Druhou část představuje naprogramování algoritmu syntézy - návrh regulátoru. Adaptivní řídicí systém je pak představován spojením obou programových modulů tedy průběžné identifikace a následné syntézy.

Práce spočívala v navržení funkčních částí programu pro PLC TC515. Dále pak v implementaci odladěných algoritmů identifikace a syntézy, v syntaxi jazyka JLR, pro tento řídicí systém.

5.3.1. Struktura programu

Logickou strukturu programu jsem volil co nejjednodušší s využitím procesů uživatelského programu P0, P5 nebo obou (viz 2.1.2). Program je dále dělen do dílčích funkčních bloků, které jsou pak podle potřeby řazeny za sebe do výpočetního řetězce.

5.3.2. Vzorkovací perioda

Jelikož jsem neměl praktické zkušenosti s programováním PLC automatů v prostředí Epos, volil jsem nejprve hodnotu 400ms, která je programově realizována použitím procesu P5. Dále se ukázalo, že je možné velikost vzorkovací periody zmenšit na 100ms. Tato vzorkovací perioda je realizována programově využitím systémového časovače.

Volba velikosti vzorkovací periody Ts souvisí také se strukturou programu. Obecná perioda, s kterou jsou vyhodnocovány vstupy PLC je daná dobou cyklu programu. Tato doba především závisí na délce a struktuře uživatelského programu. Zaručení konstantní vzorkovací periody Ts, se kterou bude systém číst z AD převodníku, vykonávat výpočetní algoritmus a zapisovat na DA převodník, je nutné programově ošetřit. Lze to provést tedy dvěma způsoby. Prvním je využití jednoho z časově aktivovaných procesů. V mém případě to je proces P5, který je vykonáván vždy po 400ms. Pak pouze stačí výpočetní algoritmus umístit do tohoto procesu. Druhým způsobem je využití základního procesu P0. Tento proces je prováděn vždy v každém cyklu PLC. Testováním časovače TON, který je nastaven na dobu Ts, se rozhoduje, zda bude výpočetní algoritmus umístěný v procesu P0 proveden, nebo přeskočen. V případě že je přeskočen, bude PLC vykonávat jen prázdné cykly až do naplnění doby Ts v časovači TON.

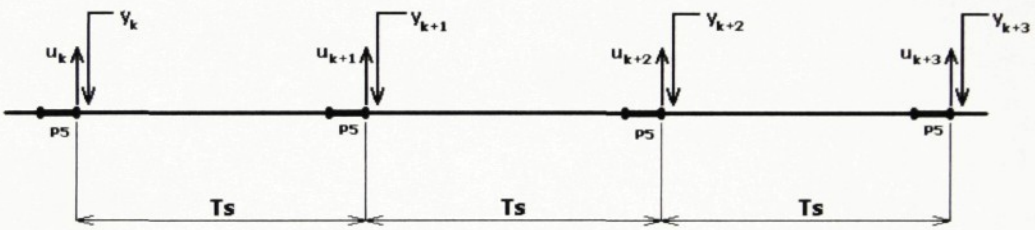
Obě dvě možnosti jsem odzkoušel a podle potřeby jsem těmito způsoby volil vzorkovací periodu 400ms, 200ms a 100ms. Délka vzorkovací periody Ts je pak dána strukturou a vlastnostmi časovače TON.

5.3.3. Cykly programu

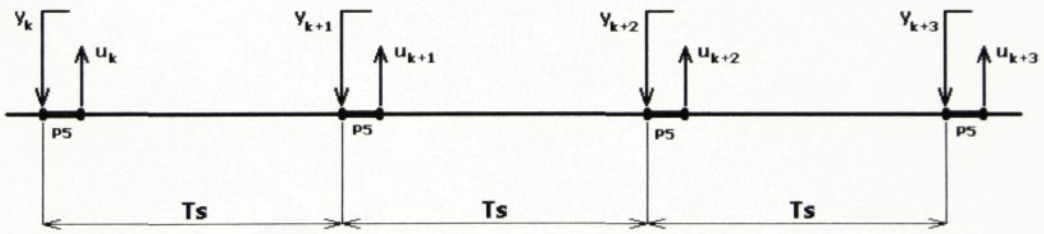
Cyklickým vykonáváním uživatelského programu PLC automatu, představujícím jednotlivé operace nazvané souhrnně cyklem programu (obr.2.1), jsou v technické

dokumentaci [7] jednou označovány činnosti řešení programu, aktualizace výstupních proměnných do výstupních periferních jednotek a aktualizace stavů ze vstupních periferních jednotek, podruhé pak čtení stavů ze vstupních periferních jednotek, řešení programu a nakonec zápis výstupních proměnných do výstupních periferních jednotek.

Jistá nejednoznačnost v tomto pojmu mě vedla k dvěma různým přístupům, jakým způsobem a kdy provádět čtení ze vstupního AD převodníku a zápis na výstupní DA převodník (analogový vstup AI0 a výstup AO0). Označím-li je jako varianta první a varianta druhá, potom funkci programu, při volbě varianty první, představuje obr. 5.13 a varianty druhé obr. 5.14. Obě dvě varianty bylo tedy nutno speciálním způsobem programově ošetřit.



Obr. 5.13 Varianta první



Obr. 5.14 Varianta druhá

Vzhledem k tomu, že diskrétní model dynamické soustavy získaný první variantou je znám až v dalším diskrétním časovém kroku (model je vypočten až v následujícím cyklu programu, tedy od aktuálního měření uběhlo 400ms), je použití tohoto postupu nevýhodné. Dále pak dosažení přesnějšího modelu dynamické soustavy druhou variantou, kde nedochází ke zpoždění, jednoznačně hovoří v prospěch varianty dvě.

5.3.4. Nastavení vstupní a výstupní jednotky

Pro regulaci uvažované soustavy používám v tomto programu jen analogové vstupy a výstupy jednotky PLC. Pro akční veličinu analogový výstup *AO0*, pro měření výstupní veličiny analogový vstup *AI0*. Obsluha se provádí pomocí obrazů vstupů a výstupů (proměnných) alokovaných do příslušné oblasti X nebo Y zápisníkové paměti. Přiřazením umístění určité proměnné v softwarové konfiguraci vstupní a výstupní jednotky PLC se aktivují dané vstupy. Je možné prakticky libovolně přiřazovat umístění obrazů vstupů a výstupů v zónách X, Y, R zápisníkové paměti, a tak postupně aktivovat konkrétní vstupy a výstupy při ladění programu bez nutnosti fyzického odpojování nebo připojování svorkovnic.

V prostředí Epos se softwarová konfigurace zadává pomocí chráněného slova *units*, za kterým následují deklarace jednotlivých jednotek. Deklarace každé jednotky je uvozena chráněným slovem *unit*, za kterým následuje konfigurace jednotky. Syntaxe zápisu je následující:

unit modul, jednotka, typ, počet_X, počet_Y, obraz_X, obraz_Y, aktivace_VV;

modul...je adresa modulu, ve kterém je jednotka instalována, pro řadu TC500 je vždy 0
jednotka... je adresa jednotky v modulu, pro analogové vstupy a výstupy je vždy 0
typ... je číselný kód charakterizující jednotku, pro analogové vstupy a výstupy je $D0
počet_X... je počet vstupních bytů jednotky
počet_Y... je počet výstupních bytů jednotky
obraz_X... je operand udávající, do kterých registrů jsou přenášena data ze vstupů
obraz_Y... je operand udávající, ze kterých registrů jsou přenášena data do výstupů
aktivace_VV... je kód udávající režim jednotky
tab... je parametr udávající jméno tabulky s inicializačními daty jednotky, rozsah

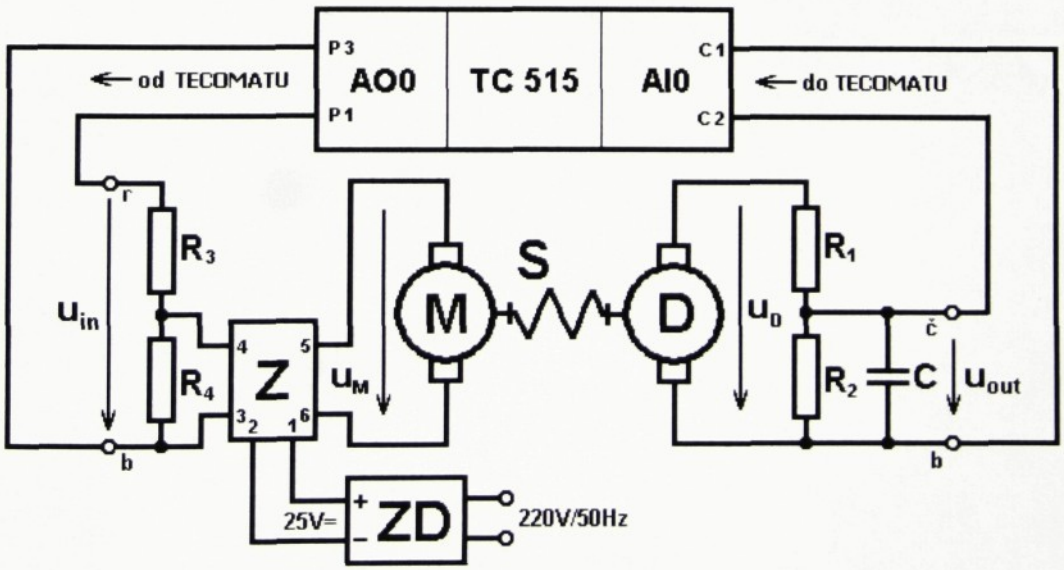
Příklad nastavení softwarové konfigurace vstupní a výstupní jednotky použitého PLC TC515 uvádím níže. Tímto zápisem se provede konfigurace prvního analogového vstupu *AI0*, binární reprezentace jeho stavu se bude přepisovat v otočce cyklu do registru s počáteční adresou X0 a jeho délka je 2 byty. Dále se pak nastaví první

analogový výstup *AO0*, jeho binární reprezentace se bude přepisovat do registru s počáteční adresou Y0 a má také délku 2 byty. Napěťový rozsah analogového vstupu bude 0-10V a oba budou aktivovány. Chráněnému slovu *unit* musí předcházet deklarace inicializační tabulky, chráněné slovo *tables*, pro nastavení rozsahu analogového vstupu.

```
tables   T1: table[1] of byte = 00000000;
unit     0, 0, $d0, 2, 2, %X0, %Y0, On, @T1;
```

Původním záměrem bylo použít pro regulaci uvažované soustavy PLC typu TC515. Jelikož se ukázalo, že 127 možných úrovní napětí na výstupu 8-mi bitového DA převodníku není dostačující (viz.5.5.) bylo vhodné použít typ TC606, vybavený 12-ti bitovým výstupním převodníkem, kterým je na zakázku tento model (TXK 082 60 v.č. K90063) opatřen. Jedná se o zákaznickou úpravu zapojení DA převodníku.

Dále bylo potřeba upravit původní koncepci zapojení, kdy byla používána jen polovina možného napěťového rozsahu obou převodníků a začít pracovat standardně s celým rozsahem 0 až 10V. Modifikované schéma se zařazeným napěťovým děličem 1:2 na vstupu regulované soustavy a vhodně upraveným děličem na výstupu soustavy je na obr.5.15. Toto zapojení platí pro oba typy PLC, pouze označení svorek analogového výstupu PLC TC606 je P1 a P2.



Obr.5.15 Modifikované zapojení regulované soustavy a řídicího systému TECOMAT

Popis signálů:

u_{in} ... řídící napětí z TC515, 0-10V

u_M ... napájecí napětí motoru, 0-24V

u_D ... napětí na tachodynamu, 0-24V

u_{out} ...vstupní napětí do TC515, 0-10V

Hodnoty odporů:

R_1 ... 10,0 KΩ

R_2 ... 10,2 KΩ

R_3 ... 5,0 KΩ

R_4 ... 5,1 KΩ

Příklad nastavení softwarové konfigurace vstupní a výstupní jednotky pro PLC TC606 uvádím zde.

```
tables    T1: table[1] of byte = 00000000;
unit 0, 0, $D0, 2, 0, %X0, On, @T1;
```

Tento zápis provede konfiguraci pouze prvního analogového vstupu *AI0*, se stejnými parametry jako v předchozím případě. Pro nastavení upraveného analogového výstupu *AI0* není možné použít tuto standardní softwarovou konfiguraci pomocí direktivy *units*. Obsluhu tohoto DA převodníku zprostředkovává speciální USI instrukce dodaná k tomuto PLC. Nejdříve je nutné tuto tzv. uživatelskou USI instrukci deklarovat pomocí chráněného slova *usis*. Pro použití této instrukce je vhodný proces P64, je to závěrečný proces cyklu programu, který se provádí vždy jako poslední ze všech vykonávaných procesů aktuálního cyklu. Samotný zápis instrukce se provádí funkcí *setusi(jméno uživatelské instrukce)*. Proměnná obsahující číslo, které se má zapsat na DA převodník se musí načíst jako *index*. Na akumulátor se tedy zapíše adresa proměnné od začátku registru R pomocí mnemonické instrukce čtení dat na vrchol zásobníku.

Příklad deklarace uživatelské instrukce pro obsluhu 12-ti bitového DA převodníku a zápis instrukce čtení dat je zde.

```
usis      OT13_12B = Ot13_12b.uid;

ld  @.Výstup;
setusi(OT13_12B);
```

Výsledný navržený program adaptivního regulátoru je tedy proveden ve dvou variantách. Pro PLC TC515 a PLC TC606. Rozdíl je pouze v tom, že na PLC TC606 program obsluhuje 12-ti bitový výstupní převodník.

Tedy oba programy pracují stejným způsobem, rozdíl je pouze v číselném rozsahu akční veličiny. Navíc PLC TC515 má naprogramován operační panel s LCD displejem pro komunikaci s uživatelem. Dále pak obě varianty programu komunikují s navrženou vizualizační aplikací, která slouží jako vyšší uživatelské rozhraní.

5.3.5. Popis programu

Vytvořený program Adapt12bit.jlr realizující funkci adaptivního regulátoru je zapsán v syntaxi jazyka JLR, v prostředí EPOS for Windows.

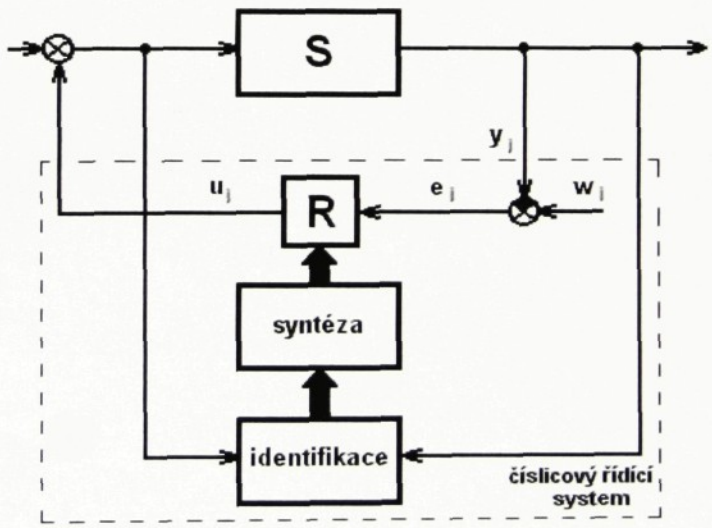
Program pracuje na PLC TC606 se vzorkovací periodou 400ms, diskrétní model soustavy je čtvrtého řádu a obsluhuje 12-ti bitový vstupní a výstupní převodník s napěťovým rozsahem 0-10V.

Samotný program se skládá ze dvou hlavních částí, deklarační a příkazové. V deklarační části programu jsou uvedeny deklarace všech nezbytných konstant, typů a proměnných a dále také definice tabulky popisující napěťové rozsahy vstupního 12-ti bitového AD převodníku, definice vstupních a výstupních jednotek PLC TC606 (adresa, typ jednotek, jejich počet a obrazy v zápisníkové paměti) a deklarace uživatelské instrukce USI pro zápis na výstupní 12-ti bitový DA převodník.

Příkazová část programu se skládá z jednotlivých uživatelských procesů. Základní proces P0 není v tomto případě naprogramován, ale musí být uveden. Časově aktivovaný proces P4 obsahuje jednotlivé funkční bloky výpočetních algoritmů. Jsou to: *Command Init* - počáteční inicializace jednotlivých proměnných programu, *Command Reg* - výpočet akčního zásahu, *Command Sample* - vzorkování, příprava a manipulace s diskrétními časovými řadami hodnot výstupní veličiny y a akční veličiny u, *Command Steady* - výpočet střední hodnoty výstupní veličiny y v počátečním pracovním bodě v ustáleném stavu, *Command Ident* - algoritmus LD-filtru pro identifikaci dynamické soustavy, *Command Dyn* - algoritmus dynamického programování pro návrh optimálních parametrů diskrétního regulátoru při minimalizaci kvadratického kritéria.

Tyto bloky jsou podle potřeby zařazovány do těla programu vyhodnocením podmiňovacího příkazu *if* a příslušné řídící proměnné typu *bool*. Dále se v příkazové části programu nacházejí procesy teplého a studeného restartu P62 a P63. Nakonec je to závěrečný proces cyklu programu P64, který obsahuje zápis mnemonické instrukce čtení dat a dále také zápis uživatelské instrukce USI pro obsluhu 12-ti bitového DA převodníku.

Základní struktura adaptivního řídícího obvodu je na obr.5.16. Zařazením pomyslných vypínačů do obvodu, které jsou představovány programově jednotlivými řídícími proměnnými, lze realizovat různé činnosti adaptivního systému. Nejdříve lze spustit identifikaci v otevřené zpětnovazební smyčce, poté provést syntézu - návrh optimálních parametrů regulátoru a nakonec uzavřít regulační smyčku. Identifikace a syntéza může být vypnuta a systém pak pracuje se seřízeným regulátorem. Dále je možné nastavit i režim řízení s průběžnou identifikací, kdy dochází k průběžnému odhadu parametrů regulované soustavy.



Obr.5.16 Struktura adaptivního řídícího obvodu

5.3.6. Uživatelské rozhraní a ovládání programu

Pro snadnější ovládání programu a sledování regulačních pochodů v reálném čase, jsem vytvořil uživatelské rozhraní v prostředí pro tvorbu vizualizací aplikací řídicích systémů TECOMAT Reliance.

Programovatelný logický automat funguje jako autonomní systém a k samotné činnosti nepotřebuje počítač PC. Ten funguje jako nadřazený systém vývojového prostředí a může sloužit i jako terminál pro monitorování řízeného technologického procesu. Základní interakci s uživatelem zprostředkovává operační panel, který je tvořen dvouřádkovým LCD displejem. V případě PLC TC606 se jedná o přídavné zařízení ID_04, u PLC TC515 je operační panel integrován přímo na automatu. Pro širší demonstraci programu slouží vytvořená aplikace v prostředí Reliance.

Reliance verze 2.5.5 je představen jako komplexní profesionální SCADA/MMI systém určený k tvorbě aplikací pro monitorování, řízení a automatizaci technologických procesů v reálném čase. Je vytvořen jako 32bitová aplikace určená pro operační systémy Windows 95/NT. Pro tvorbu aplikací poskytuje Reliance výkonné, moderní RAD (rapid application development) vývojové prostředí, které svým vzhledem a způsobem ovládání připomíná prostředí produktů firmy Borland jako jsou Delphi, C++ Builder či JBuilder. Vývojové prostředí tohoto softwaru obsahuje běžné součásti jako systém popup menu, nástrojové lišty, objekt inspektor, paleta připravených komponent, horké klávesy a další prvky.

Vývoj aplikace probíhá v tomto prostředí poměrně intuitivně. Postupoval jsem podle dostupné dokumentace [16] a v relativně krátké době se mi podařilo vytvořit celou aplikaci. Používal jsem Lite verzi Reliance 2.5.5 omezenou na maximální počet 25 proměnných. Jedná se o volně šířenou verzi, jinak plně funkční, přesto bylo bohužel i pro mojí malou aplikaci toto omezení na hranici únosnosti.

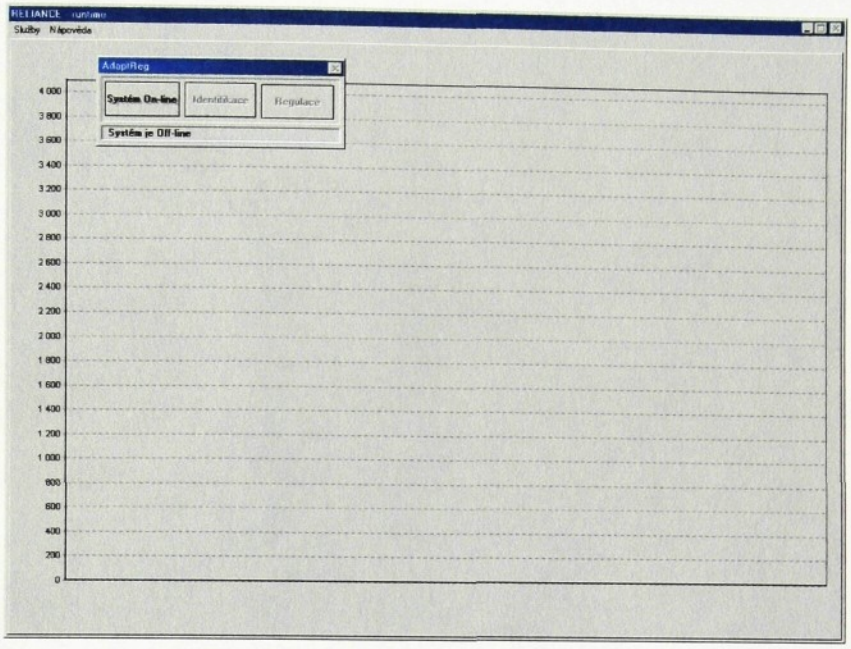
Před návrhem samotných formulářů (jednotlivých oken) vyvíjené aplikace, je nutné nastavit celou řadu parametrů a vytvořit celkovou strukturu projektu. Po založení nového projektu je prvním krokem nadefinování parametrů PLC, s kterým bude aplikace komunikovat. Dále je to nadefinování automaticky vyčítaných zón paměti PLC. Poté vložení příslušných proměnných z těchto zón do projektu. Druhým krokem je vytvoření funkčního celku představující jedno dispečerské stanoviště a definice PC, které do tohoto celku patří a připojení nadefinovaného PLC. Třetím krokem je nastavení parametrů pro start aplikace v modulu Runtime. Nakonec přichází na řadu tvorba vlastní vizualizace. Do formulářů se postupně umísťují různé komponenty (tlačítka, displeje, bitové mapy) a jednotlivým komponentám se nastavují jejich vlastnosti. Dále se doplní

ošetření událostí (například stisk tlačítka). Vizualizační prvky se pak navážou na dříve definované proměnné, jejichž hodnota je buď zobrazována, nebo má vliv na způsob zobrazení prvku. Nadefinují se databázové soubory, do kterých jsou vybrané hodnoty ukládány. Nad těmito databázemi se definují uživatelské grafy nebo tabulkové sestavy, které umožňují zobrazení dat z archivů.

Po spuštění aplikace v runtime modulu (obr.5.17) dojde nejdříve k inicializaci celého projektu, navázání komunikace s PLC přes komunikační driver a nastavení vyčítaných zón zápisníkové paměti automatu. O těchto akcích informuje okno systémových informací.

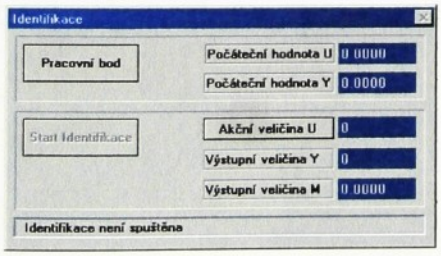
Obr. 5.17 Start aplikace

Poté se objeví základní okno uživatelského rozhraní (obr.5.18), které slouží jako lišta pro přepínání mezi dalšími okny. Těmi jsou okna Identifikace a Regulace. Pracovní plochu tvoří okno Grafu, ve kterém se v reálném čase vykreslují průběhy výstupní veličiny y a akční veličiny u. Základní okno obsahuje tlačítka Systém On-line, Identifikace a Regulace, dále pak stavový pruh, který informuje zda je program aktivní či ne. Pokud není program aktivní, nelze volit ani jedno z tlačítek. V opačném případě se stiskem tlačítka Identifikace dostaneme do příslušného okna (obr.5.19).

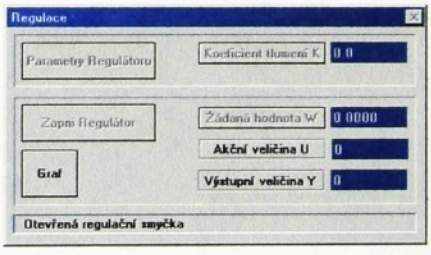


Obr. 5.18 Základní okno aplikace

Jeho obsah je rozdělen do dvou částí. V první části je tlačítko Pracovní bod, po jehož stisku regulovaná soustava najede do pracovního bodu a program si označí aktuální hodnoty akční a výstupní veličiny jako počáteční. V druhé části je tlačítko Start identifikace po jehož stisknutí začne program identifikovat připojenou soustavu. Nyní je nutné zadat skokové změny akční veličiny *u*. Stavový pruh na spodu okna opět informuje zda probíhá identifikace soustavy, či nikoli.



Obr. 5.19 Okno Identifikace



Obr. 5.20 Okno Regulace

Okno Regulace má opět dvě části (obr.5.20). V první je tlačítko Parametry regulátoru, po jehož stisknutí program začne navrhovat parametry diskrétního regulátoru. Dále je zde možné zadat koeficient tlumení *Kapa*. V druhé části je tlačítko Zapni regulátor, po jehož stisknutí dojde k uzavření regulační smyčky a řízení na

žádanou hodnotu w. Její velikost lze zadat po stisku tlačítka Žádaná hodnota. Stavový pruh ve spodní části okna informuje o uzavření nebo otevření regulační smyčky.

Všechna tlačítka jsou ošetřena tak, že je lze volit až po provedení předchozích logicky souvisejících operací. Grafické provedení je tzv. třístavové s aretací, tzn. že je definováno jak má vypadat tlačítko ve chvíli, kdy nad ním přejíždíme myší. Dále je také na první pohled vidět, zda je funkce vybrána či ne (text tlačítka se mění podle toho zda je v poloze zapnuto nebo vypnuto).

5.4. Výsledky identifikace soustavy

Funkci navrženého programu, pro identifikaci regulované soustavy, jsem ověřoval na PLC TC515, připojeného k uvažované regulované soustavě dle obr.3.4.

Při identifikaci soustavy uvažujeme otevřenou zpětnovazební smyčku a přímé řízení akční veličinou. Identifikace je spuštěna v uvažovaném pracovním bodě regulované soustavy z ustálené hodnoty výstupní veličiny y. Poté následují skokové změny akční veličiny u. V pracovním bodě je potřeba vytvořit nulové počáteční podmínky. Od skutečných hodnot výstupní a akční veličiny, které vstupují do identifikačního algoritmu jsou odečteny jejich ustálené hodnoty v tomto pracovním bodě. Platí tedy vztahy (5-11) a (5-12), kde y_m, u_m jsou aktuální měřené hodnoty a y_p, u_p jsou hodnoty v uvažovaném pracovním bodě.

$$y = y_m - y_p \tag{5-11}$$

$$u = u_m - u_p \tag{5-12}$$

Nejprve bych uvedl výsledky identifikace provedené programem Identf1.jlr. Program pracuje ve smyslu první varianty (viz.5.3.3.). Vzhledem k tomu, že program byl navržen pro diskrétní model jak třetího, tak i čtvrtého řádu, uvádím oboje výsledky.

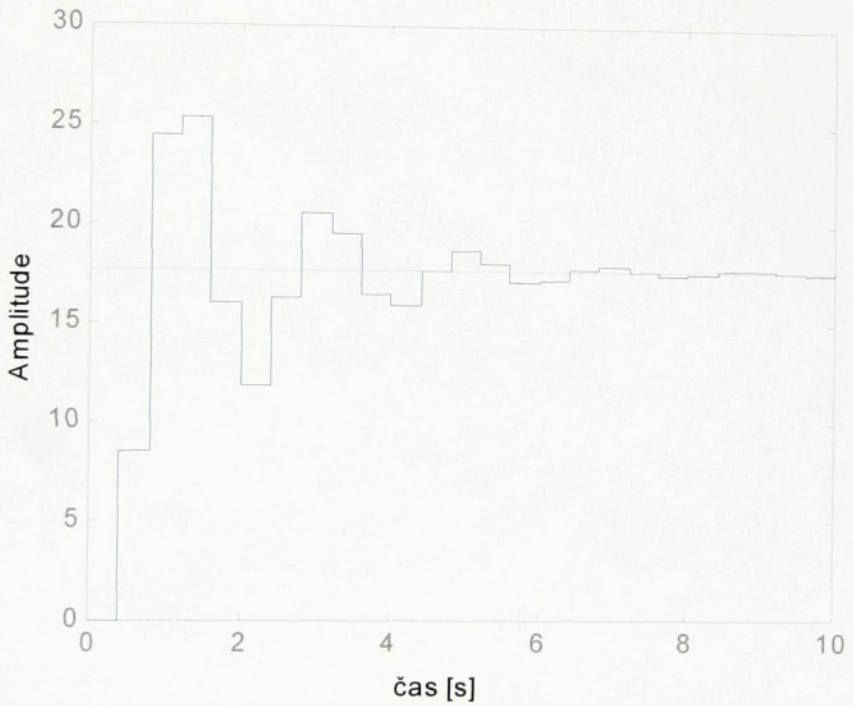
Obrazový přenos diskrétního modelu třetího řádu:

$$F(z) = \frac{z^{-1}\left(b_0 + b_1 z^{-1} + b_2 z^{-2}\right)}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}} \tag{5-13}$$

Výsledný obrazový přenos diskrétního modelu třetího řádu regulované soustavy, vypočtený programem Identf1.jlr:

$$F_{3v1}(z) = \frac{z^{-1}\left(8{,}5381 + 6{,}4119z^{-1} - 8{,}8843z^{-2}\right)}{1 - 1{,}1082z^{-1} + 0{,}9043z^{-2} - 0{,}4527z^{-3}} \tag{5-14}$$

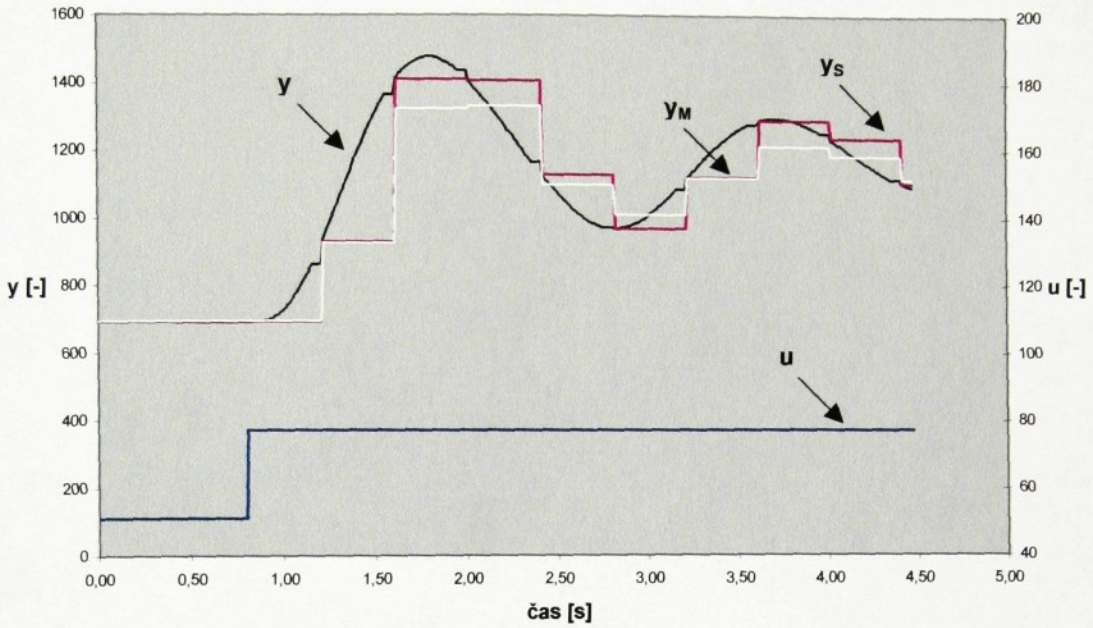
Přechodová charakteristika tohoto diskrétního modelu regulované soustavy vykreslená pomocí funkce step je na obr.5.21.



Obr.5.21 Přechodová charakteristika diskrétního modelu třetího řádu F_{3V1}

K naměření průběhů výstupní, akční veličiny a simulované odezvy diskrétního modelu jsem používal tzv. systémový analyzátor (viz.2.2.2). Ve skutečnosti jsou tyto veličiny představovány jednotlivými proměnnými programu.

Grafické znázornění výsledků identifikace z naměřených hodnot systémovým analyzátorem je na obr.5.22. Je nutné si uvědomit, že diskrétní model regulované soustavy je v případě varianty první zpožděn o jeden diskrétní časový krok. Z důvodu porovnání je ale odezva vypočteného modelu posunuta v grafu o jeden krok zpět.



Obr.5.22 Výsledky identifikace programem Identf1.jlr: naměřené hodnoty výstupní veličiny y, navzorkované hodnoty výstupní veličiny y_S, odezva identifikovaného diskrétního modelu regulované soustavy y_M, skok akční veličiny u

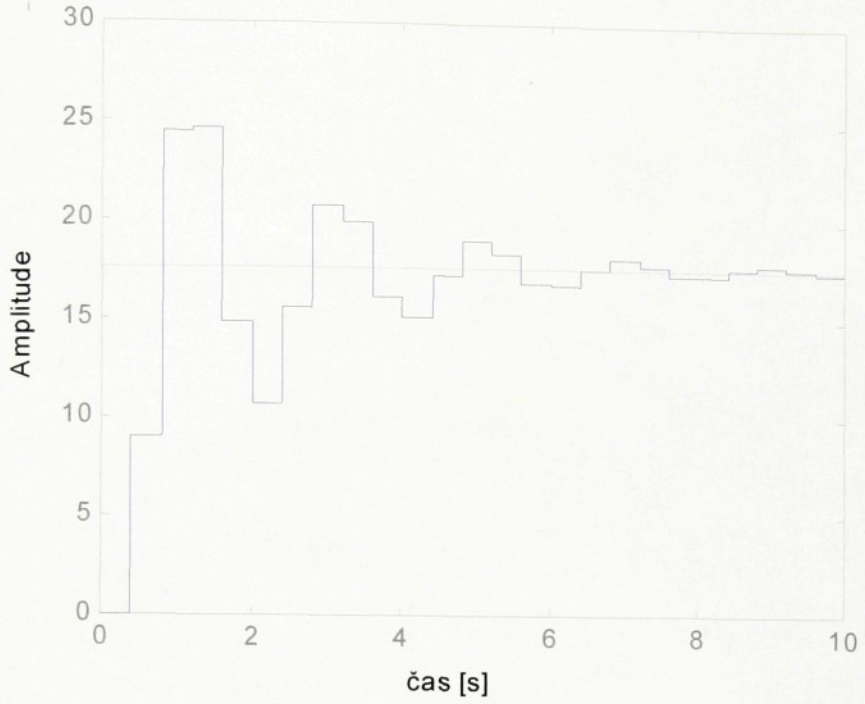
Obrazový přenos diskrétního modelu čtvrtého řádu:

$$F(z)=\frac{z^{-1}\left(b_0+b_1z^{-1}+b_2z^{-2}+b_3z^{-3}\right)}{1+a_1z^{-1}+a_2z^{-2}+a_3z^{-3}+a_4z^{-4}} \tag{5-15}$$

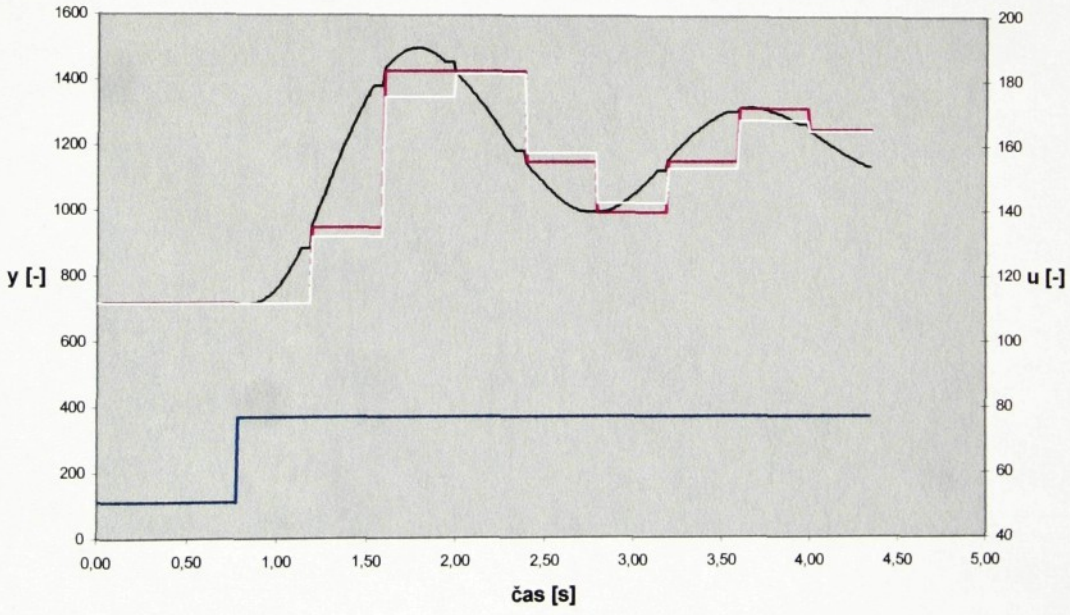
Výsledný obrazový přenos diskrétního modelu čtvrtého řádu regulované soustavy, vypočtený programem Identf1.jlr:

$$F_{4v1}(z)=\frac{z^{-1}\left(8{,}9361+6{,}5760z^{-1}-8{,}7953z^{-2}-0{,}9730z^{-3}\right)}{1-1{,}0032z^{-1}+0{,}7456z^{-2}-0{,}2968z^{-3}-0{,}1195z^{-4}} \tag{5-16}$$

Přechodová charakteristika tohoto diskrétního modelu regulované soustavy vykreslená pomocí funkce step je na obr.5.23. Grafické znázornění výsledků identifikace z naměřených hodnot systémovým analyzátorem je na obr.5.24



Obr.5.23 Přechodová charakteristika diskrétního modelu čtvrtého řádu F_{4V1}



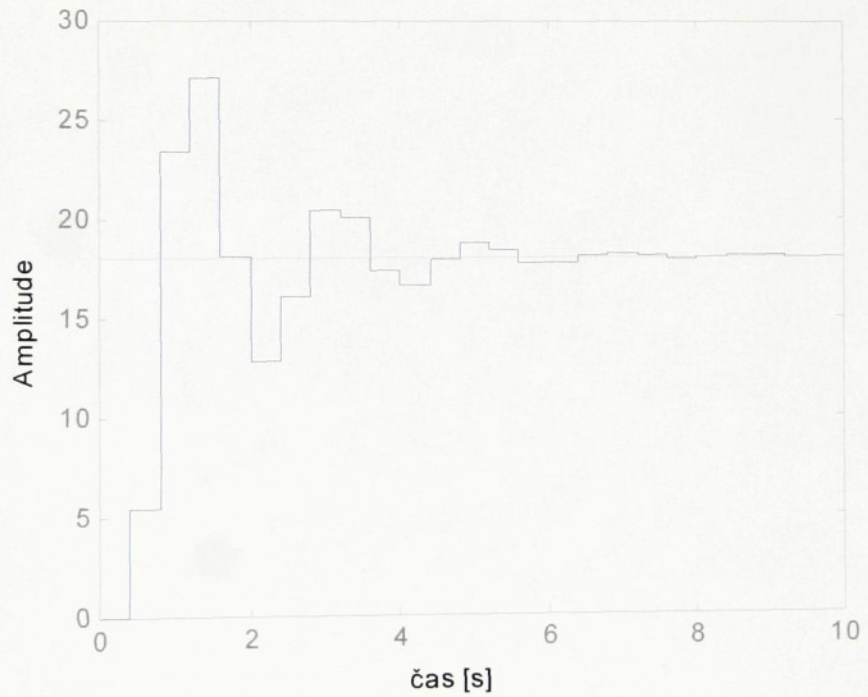
Obr.5.24 Výsledky identifikace programem Identf1.jlr: naměřené hodnoty výstupní veličiny y, navzorkované hodnoty výstupní veličiny y_S, odezva identifikovaného diskrétního modelu regulované soustavy y_M, skok akční veličiny u

Dále bych uvedl výsledky identifikace regulované soustavy provedené programem Identf2.jlr. Tento program pracuje ve smyslu varianty druhé (viz.5.3.3.). Diskrétní model regulované soustavy je opět navržen třetího i čtvrtého řádu.

Výsledný obrazový přenos diskrétního modelu třetího řádu regulované soustavy, vypočtený programem Identf2.jlr:

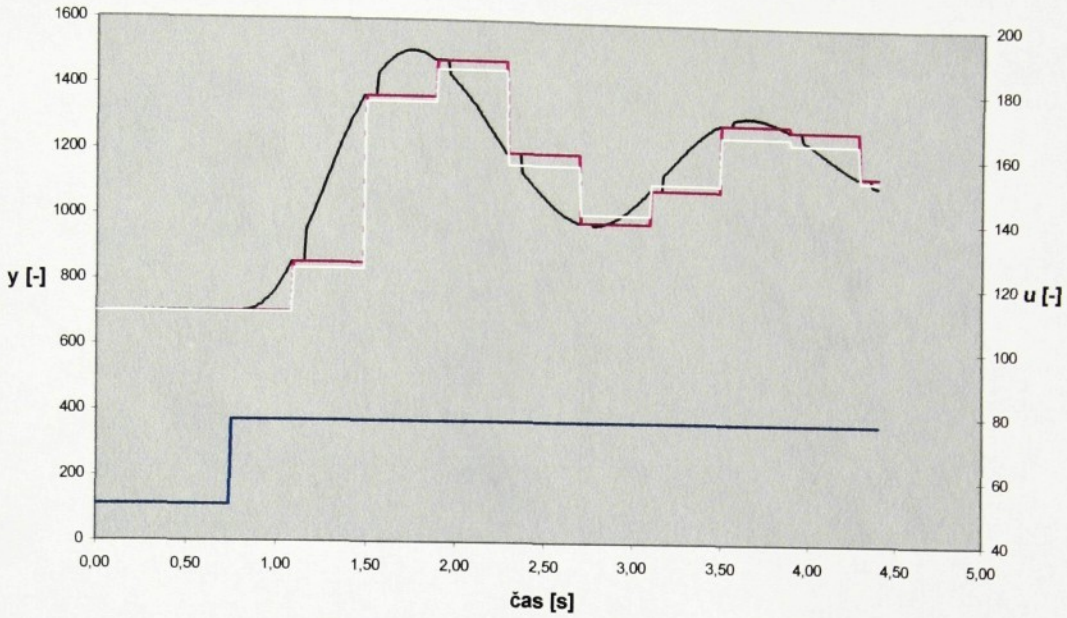
$$F_{3v2}(z)=\frac{z^{-1}\left(5{,}4808+14{,}1507z^{-1}-4{,}7778z^{-2}\right)}{1-0{,}6874z^{-1}+0{,}6995z^{-2}-0{,}1903z^{-3}} \tag{5-17}$$

Přechodová charakteristika tohoto diskrétního modelu regulované soustavy vykreslená pomocí funkce step je na obr.5.25.



Obr.5.25 Přechodová charakteristika diskrétního modelu třetího řádu F_{3V2}

Grafické znázornění výsledků identifikace z naměřených hodnot systémovým analyzátorem je na obr.5.26.



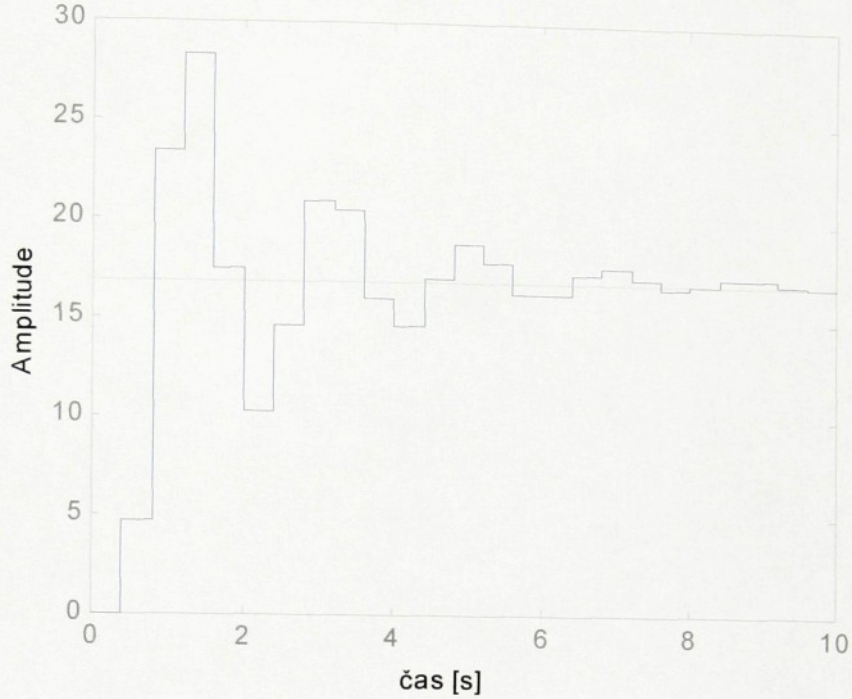
Obr.5.26 Výsledky identifikace programem Identf2.jlr: naměřené hodnoty výstupní veličiny y, navzorkované hodnoty výstupní veličiny y_S, odezva identifikovaného diskrétního modelu regulované soustavy y_M, skok akční veličiny u

Výsledný obrazový přenos diskrétního modelu regulované soustavy čtvrtého řádu, vypočtený programem Identf2.jlr:

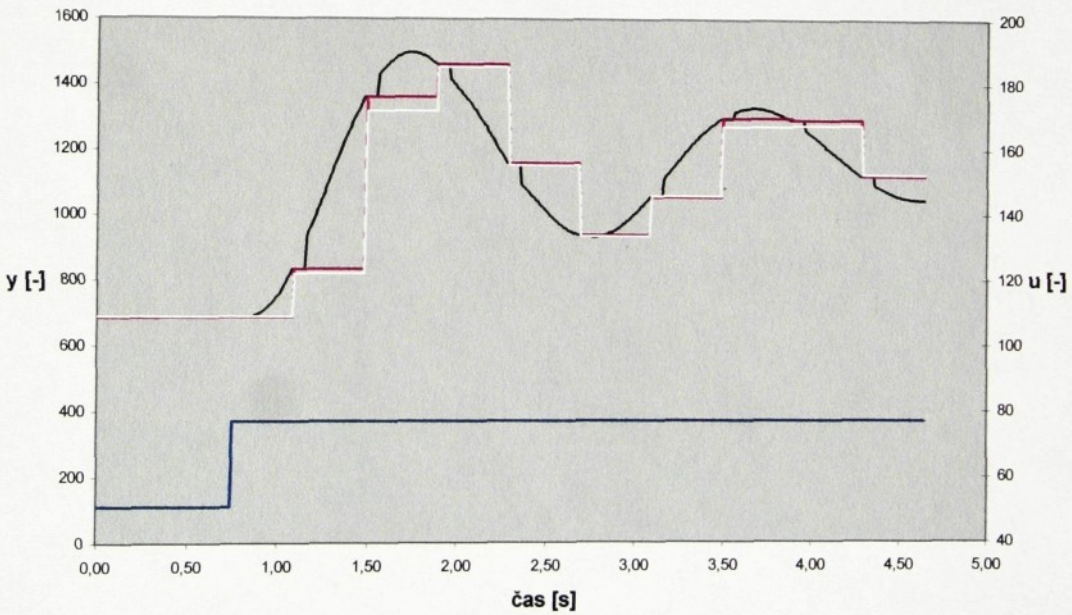
$$F_{4v2}(z) = \frac{z^{-1}\left(4{,}7310 + 13{,}8164z^{-1} - 11{,}4025z^{-2} - 5{,}1990z^{-3}\right)}{1 - 1{,}0347z^{-1} + 0{,}6552z^{-2} - 0{,}3402z^{-3} - 0{,}1650z^{-4}} \qquad (5\text{-}18)$$

Přechodová charakteristika tohoto diskrétního modelu regulované soustavy vykreslená pomocí funkce step je na obr.5.27. Grafické znázornění výsledků identifikace z naměřených hodnot systémovým analyzátorem je na obr.5.28.

Vzhledem k tomu, že diskrétní model regulované soustavy při použití varianty první je ve skutečnosti znám až v dalším diskrétním časovém kroku, je použití tohoto přístupu pro návrh adaptivního číslicového regulátoru nevýhodné. V případě varianty druhé je časové zpoždění zatíženo jen dobou výpočtu. Tato doba závisí na řádu navrhovaného modelu. Pro diskrétní model čtvrtého řádu je doba výpočtu přibližně 70ms. Z grafického znázornění výsledků je vidět, že diskrétní model čtvrtého řádu přesněji odpovídá parametrům regulované soustavy.



Obr.5.27 Přechodová charakteristika diskrétního modelu čtvrtého řádu F_{4V2}



Obr.5.28 Výsledky identifikace programem Identf2.jlr: naměřené hodnoty výstupní veličiny y, navzorkované hodnoty výstupní veličiny y_S, odezva identifikovaného diskrétního modelu regulované soustavy y_M, skok akční veličiny u

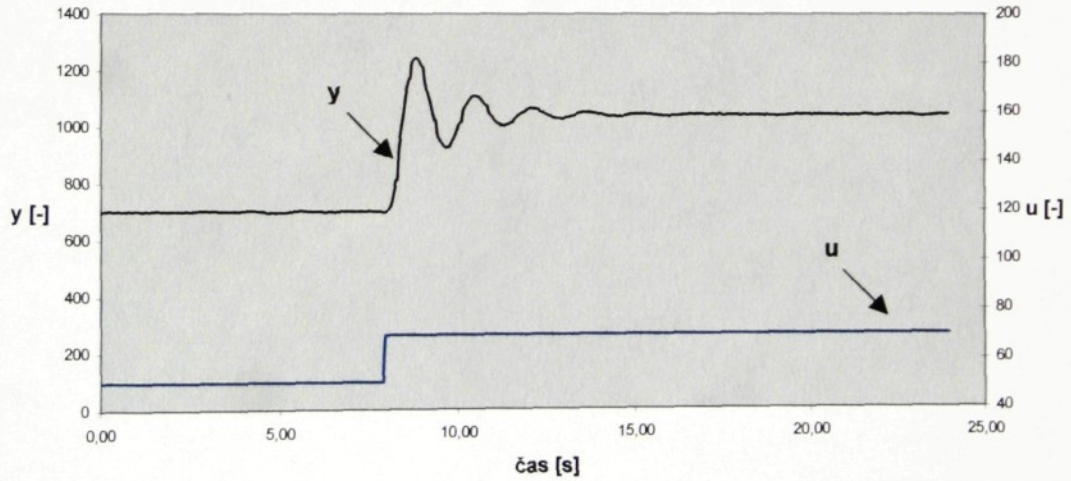
5.5. Výsledky regulace soustavy

Vlastnosti naprogramovaného adaptivního regulátoru, program Adapt8bit.jlr, jsem ověřoval na uvažované regulované soustavě, připojené k PLC TC515 podle obr.3.4.

Při zapnuté regulaci uvažujeme uzavřenou zpětnovazební regulační smyčku a řízení na žádanou hodnotu.

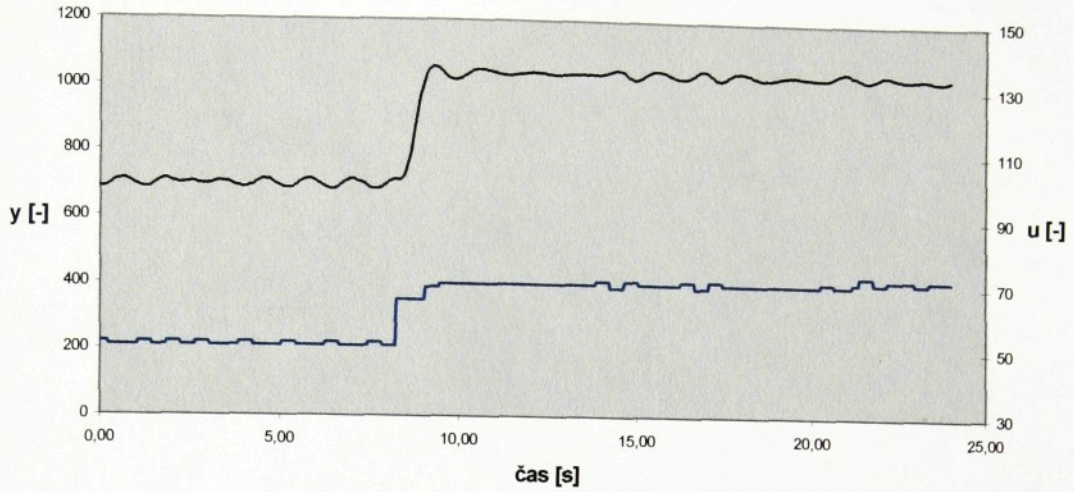
K naměření průběhů výstupní veličiny, akční veličiny a žádané hodnoty jsem opět používal systémový analyzátor (viz.2.2.2.). Všechny naměřené průběhy jsou odezvy výstupní veličiny na skokovou změnu žádané hodnoty. Jednotlivá měření jsou pro různé velikosti koeficientu tlumení regulace.

Pro porovnání uvádím na obr 5.29 odezvu výstupní veličiny na skokovou změnu akční veličiny, při vypnutém regulátoru. Měřítko a rozsah časové osy je z tohoto důvodu nastaveno stejně jako v následujících zobrazeních.

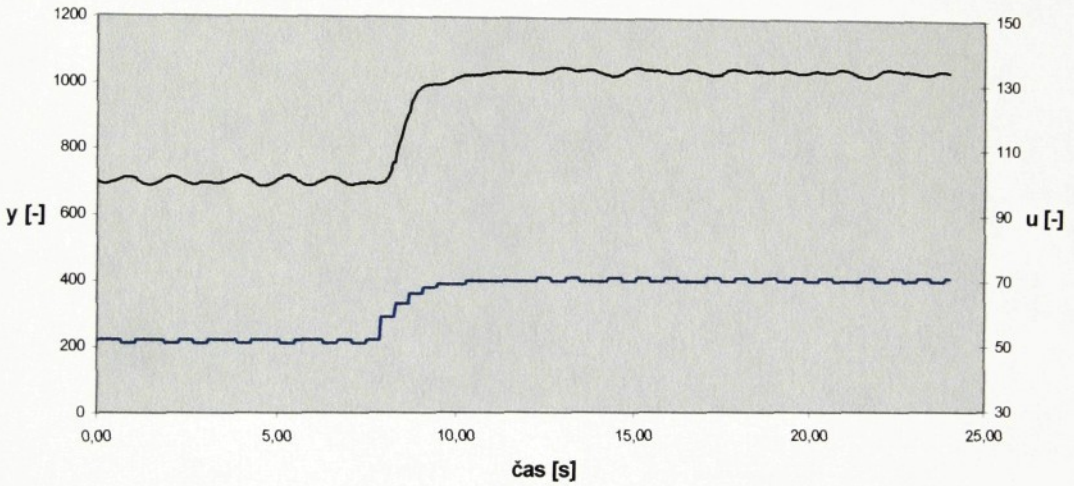


Obr 5.29 Odezva výstupní veličiny regulované soustavy y na skokovou změnu akční veličiny u

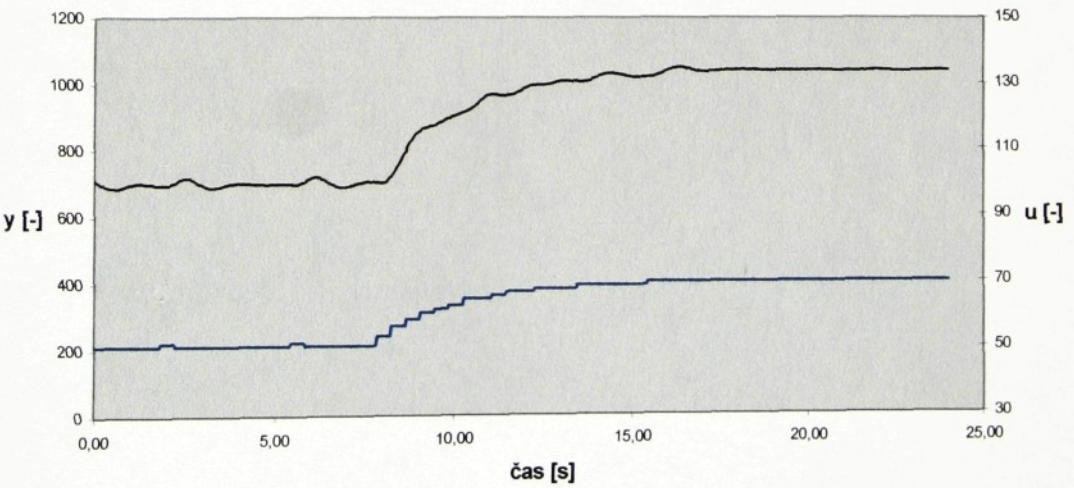
Na obr.5.30 je zobrazena odezva výstupní veličiny na skokovou změnu žádané hodnoty při koeficientu tlumení regulace 100, na obr.5.31 při koeficientu tlumení regulace 1000 a na obr.5.32 při koeficientu tlumení regulace 10000.



Obr 5.30 Odezva výstupní veličiny regulované soustavy pro K=100

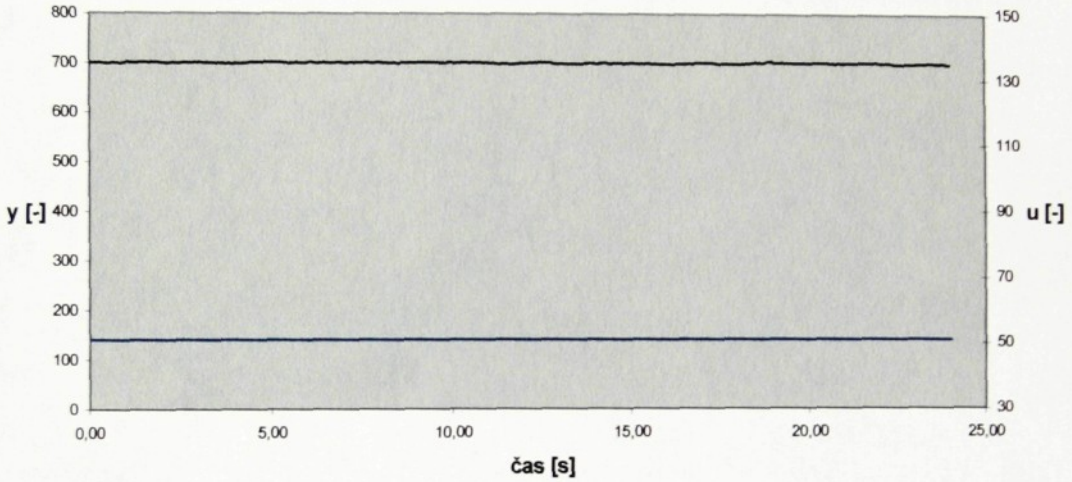


Obr 5.31 Odezva výstupní veličiny regulované soustavy pro K=1000



Obr 5.32 Odezva výstupní veličiny regulované soustavy pro K=10000

Jak je vidět, při uzavření zpětnovazební smyčky regulačního obvodu dochází k nežádoucímu kolísání výstupní veličiny. Je ověřeno, že se zvyšujícím se koeficientem tlumení regulace, tedy pro pomalejší seřízení regulátoru, se velikost kolísání zmenšuje. Tato tendence je vidět jednak na uvedených průbězích, dále pak na obr.5.33, kde je zobrazen průběh ustálené hodnoty výstupní a akční veličiny pro koeficient tlumení regulace 100000.



Obr.5.32 Průběh ustálené hodnoty výstupní veličiny pro K=100000

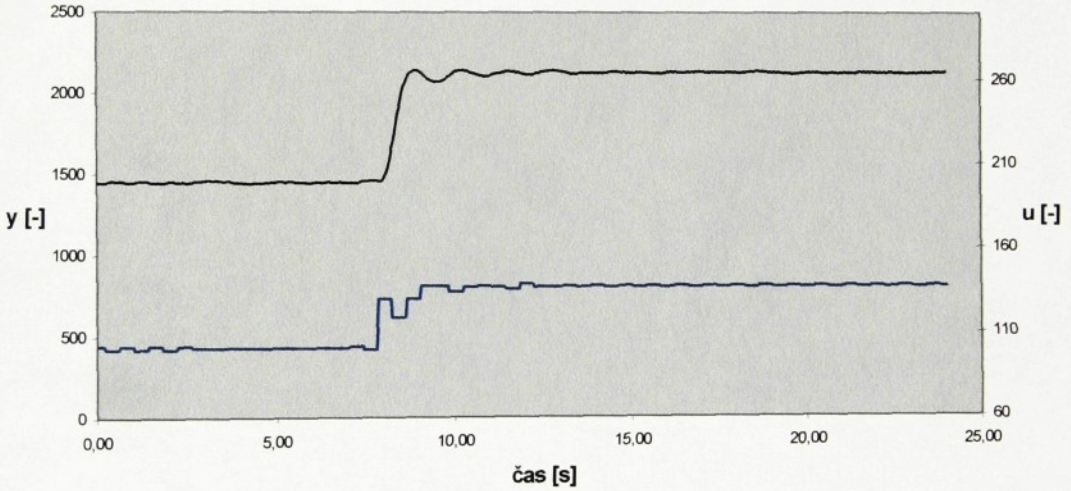
Z průběhu akční veličiny je vidět, že její hodnota se mění maximálně o jedna nahoru nebo dolů. Ve skutečnosti to však znamená poměrně velké rozvážení regulované soustavy, neboť napěťový rozsah akční veličiny na výstupním DA převodníku je používán jen z poloviny (0-5V), čemuž odpovídá při 8-mi bitovém převodu jen 127 různých napěťových úrovní akční veličiny. Znamená to, že hodnota napětí na vstupu soustavy se mění skoro o 2% a tedy při daném statickém zesílení soustavy i stejným způsobem na výstupu. Vzniká tak regulační odchylka od žádané hodnoty, kterou se regulátor snaží stále dorovnat.

Z důvodu zachování celkového zapojení regulované soustavy, byla původně volena tato koncepce.

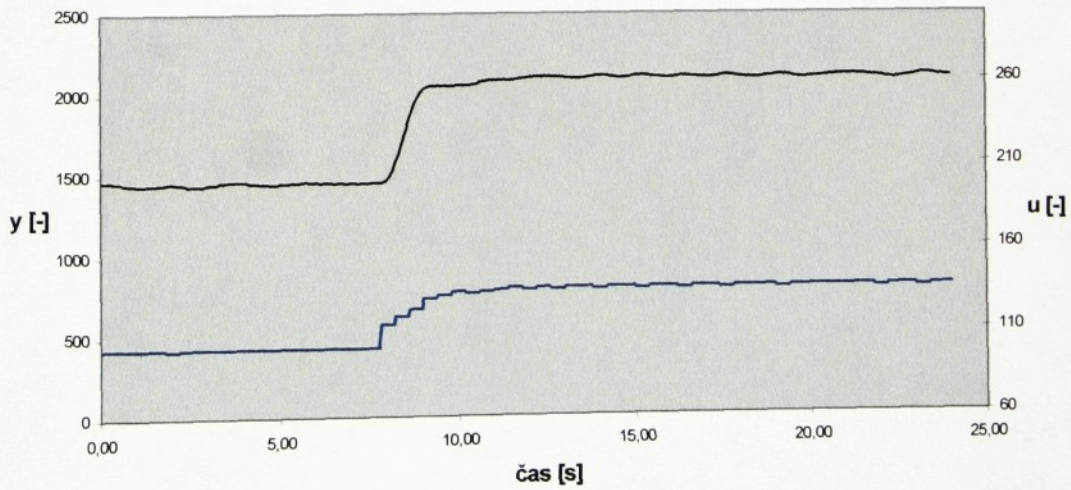
Dalším úkolem tedy bylo ověřit výše uvedené tvrzení a pokud možno odstranit toto nežádoucí kolísání výstupní veličiny regulované soustavy. Přistoupil jsem k modifikaci zapojení regulované soustavy, tak aby bylo možné využívat celý rozsah

výstupního 8-mi bitového DA převodníku a 12-ti bitového vstupního AD převodníku, jak je to běžné. Na vstup soustavy jsem tedy zařadil dělič napětí 1:2. Stejně tak na výstupu soustavy jsem dělič napětí upravil příslušným způsobem. Provedené změny jsou popsány v 5.3.5. a schéma zapojení je na obr.5.15.

Na obr.5.34 je zobrazena odezva výstupní veličiny na skokovou změnu žádané hodnoty při koeficientu tlumení regulace 100 a na obr.5.35 je zobrazena odezva výstupní veličiny na skokovou změnu žádané hodnoty při koeficientu tlumení regulace 1000.



Obr.5.34 Odezva výstupní veličiny regulované soustavy pro K=100



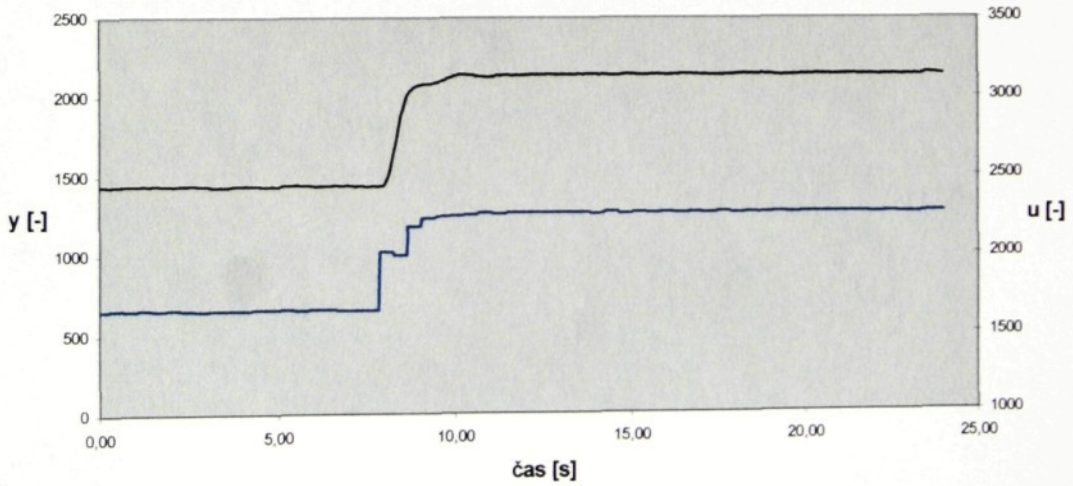
Obr 5.35 Odezva výstupní veličiny regulované soustavy pro K=1000

Při porovnání zobrazených průběhů výstupní veličiny na obr.5.30-32 s obr.5.34-35 je zřejmé, že nežádoucí kolísání výstupní veličiny regulované soustavy se opravdu zmenšilo.

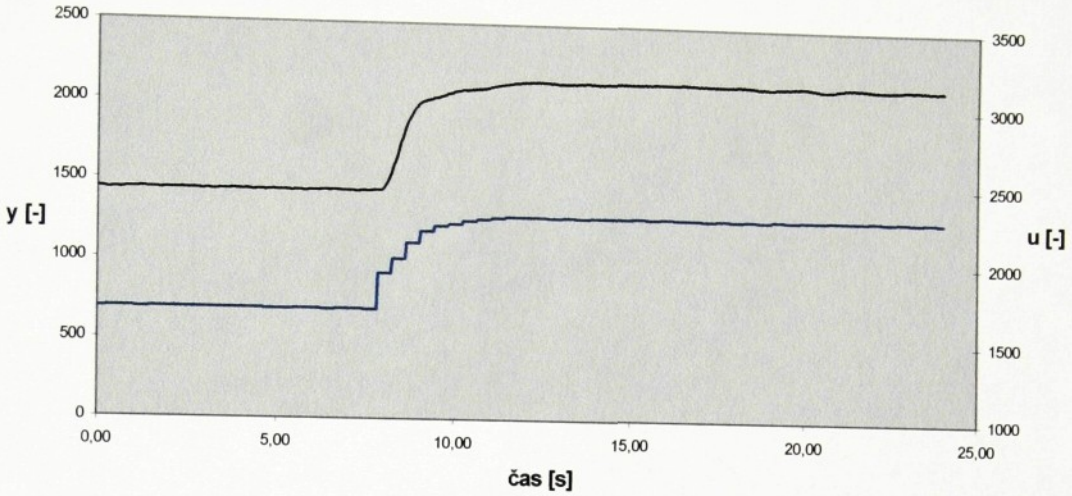
Funkci navrženého programu adaptivního regulátoru jsem dále testoval na PLC TC606, který disponuje speciálně upraveným 12-ti bitovým výstupním DA převodníkem. Výsledky regulace při použití programu Adapt12bit.jlr, jsou uvedeny níže. PLC TC606 je připojen k uvažované regulované soustavě dle obr.5.15.

Při použití 12-ti bitového výstupního DA převodníku došlo k odstranění nežádoucího kolísání výstupní veličiny regulované soustavy. Dodávané vstupní a výstupní jednotky (např. IT04 a OT04) mají běžně všechny kanály osazeny těmito 12-ti bitovými převodníky.

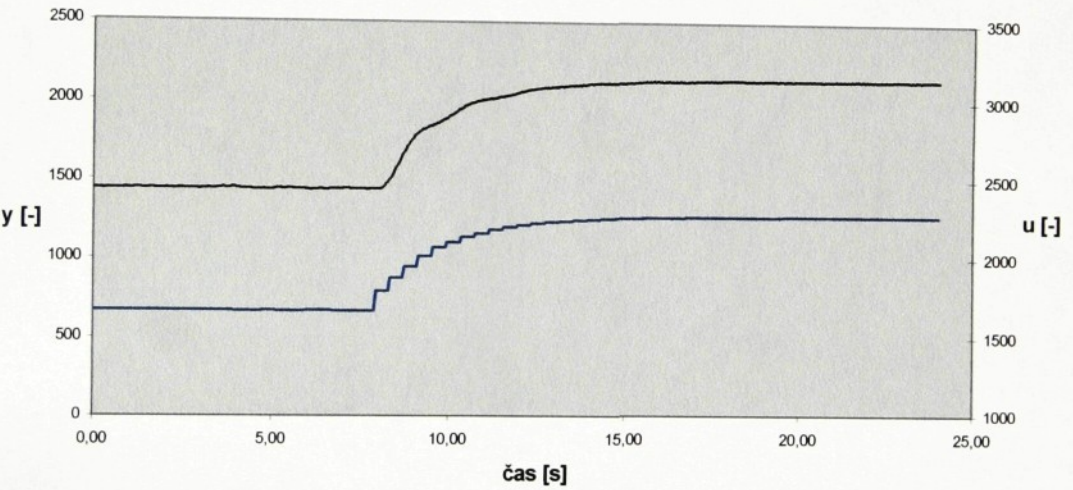
Na obr.5.36 je zobrazena odezva výstupní veličiny na skokovou změnu žádané hodnoty při koeficientu tlumení regulace 1. Na obr.5.37 je zobrazena odezva výstupní veličiny na skokovou změnu žádané hodnoty při koeficientu tlumení regulace 2 a na obr.5.38 při koeficientu tlumení regulace 25.



Obr 5.36 Odezva výstupní veličiny regulované soustavy pro K=1



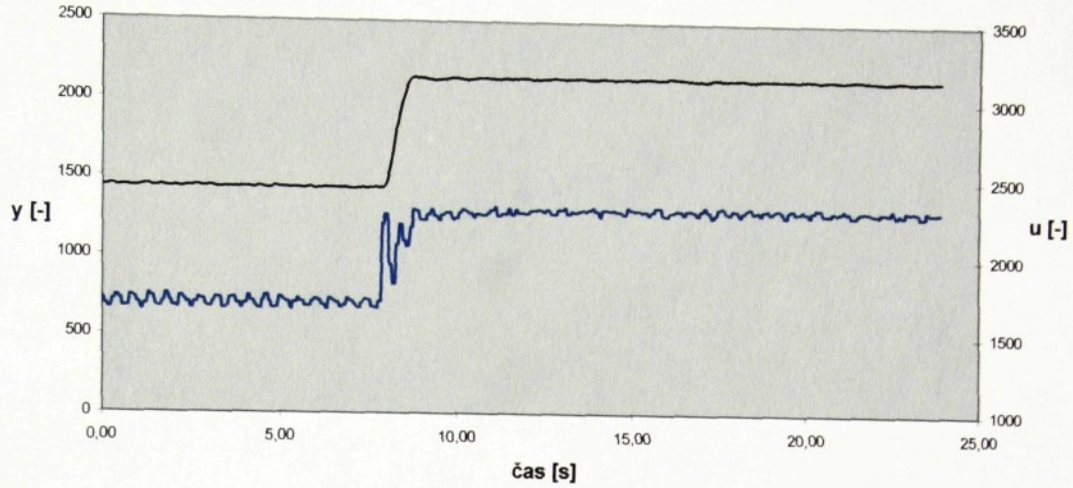
Obr 5.37 Odezva výstupní veličiny regulované soustavy pro K=5



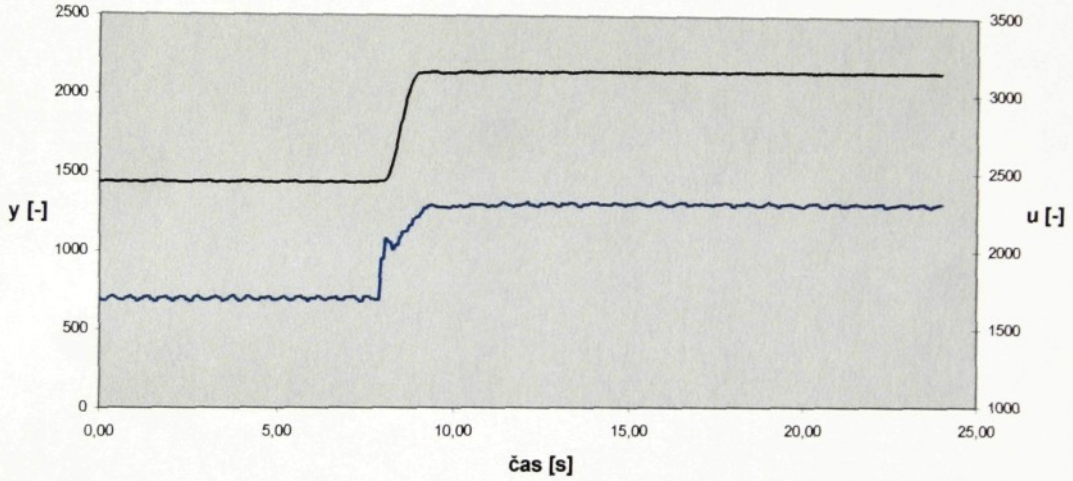
Obr 5.38 Odezva výstupní veličiny regulované soustavy pro K=25

Dále bylo vhodné ověřit chování regulované soustavy při zmenšení vzorkovací periody Ts, s kterou program pracuje. Naměřené regulační pochody při použití vzorkovací periody 100ms jsou na obr.5.39-41. Zobrazené průběhy jsou opět pro tři různé koeficienty tlumení regulace.

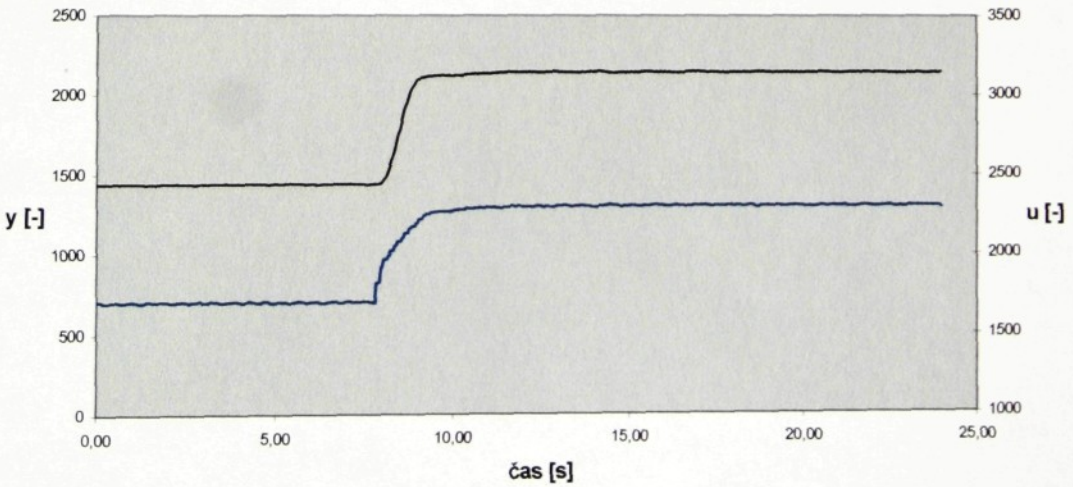
Na obr.5.39 je zobrazena odezva výstupní veličiny na skokovou změnu žádané hodnoty při koeficientu tlumení regulace 1. Na obr.5.40 je zobrazena odezva výstupní veličiny na skokovou změnu žádané hodnoty při koeficientu tlumení regulace 2 a na obr.5.41 při koeficientu tlumení regulace 25.



Obr 5.39 Odezva výstupní veličiny regulované soustavy pro K=1, T_S=100ms



Obr. 5.40 Odezva výstupní veličiny regulované soustavy pro K=5, T_S=100ms



Obr. 5.41 Odezva výstupní veličiny regulované soustavy pro K=25, T_S=100ms

Na tomto místě bych chtěl upozornit, že navržený program adaptivního regulátoru jsem dále úspěšně testoval na pomalé tepelné soustavě, která je představována průtokovým ohřívačem vody. Ovšem podrobný rozbor výsledků regulace této soustavy již není předmětem této práce.

5.6. Srovnání výsledků regulace s PID regulátorem řídicího systému TECOMAT

V této části mé práce jsem se zaměřil na porovnání dosažených výsledků regulace pomocí navrženého adaptivního regulátoru s řízením uvažované soustavy pomocí PID regulátoru.

Algoritmus PID regulátoru je již součástí programového vybavení PLC TECOMAT v podobě uzavřené instrukce s poměrně značným počtem parametrů. Pomocí instrukce PID lze řídit soustavy, u kterých je doba přechodového děje aspoň o řád delší, než je perioda vzorkování regulátoru. Vzorkovací periodu je nutné nastavit s ohledem na dobu cyklu PLC tak, aby byla zaručena určitá přesnost vzorkování. Je vhodné, aby vzorkovací perioda regulátoru byla stanovena o řád výše, než je doba cyklu PLC. K měřítkování nebo filtraci naměřených hodnot je možné použít instrukci CNV. Ta provádí také základní normalizaci unifikovaných, odporových a teplotních rozsahů a provádí základní diagnostiku měření analogových jednotek. Instrukce PID zajišťuje ve volitelných násobcích 10ms výpočet hodnoty akčního zásahu podle algoritmu PID. Algoritmus pracuje v zásadě podle diskrétní verze této rovnice.

$$u(t) = K\left[e(t) + \frac{1}{T_I}\int_0^t e(\tau)d\tau + T_D\frac{de(t)}{dt}\right] \tag{5-19}$$

Ovládání algoritmu je zajištěno pomocí struktury proměnných, která je definována na registrech PLC. Parametry regulačního algoritmu se zadávají do této rezervované datové oblasti v zóně registrů R. Nastavení všech žádaných parametrů regulátoru je vhodné provést v procesu teplého restartu P63. V průběhu regulace je pak možné měnit tyto parametry regulátoru dosazením nových hodnot do patřičných proměnných. Dále je nutné v průběhu regulace dosazovat do příslušné proměnné *Input1*

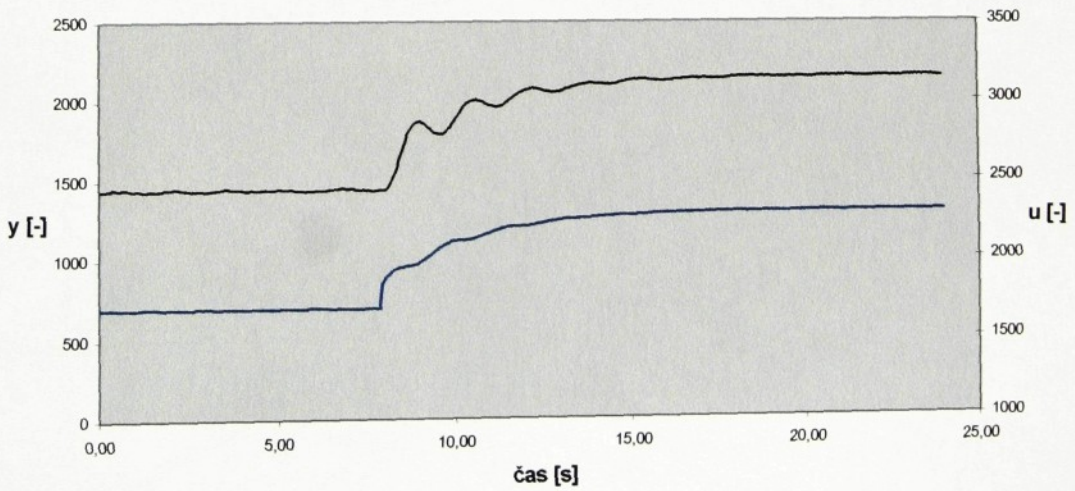
hodnotu regulované veličiny a v základním procesu P0 je nutné před použitím instrukce PID naplnit zásobník indexem registru počátku datové struktury s jednotlivými parametry. Po výpočtu pak stačí přepsat hodnotu vypočteného akčního zásahu z proměnné *ConOut* do analogové výstupní jednotky.

Jednotlivé parametry jsou velmi podrobně popsány v [11], omezím se tedy pouze na jejich základní popis a na hodnoty, které jsem jim přiřadil. Proměnná *MinY=0 a MaxY=10000* je minimální a maximální měřená hodnota. Používá se pro normalizaci odchylky. Měřená (regulovaná) veličina je deklarována proměnnou *Input1*. Proměnná *gW* je žádaná hodnota, ležící v intervalu měřené veličiny. Současnou žádanou hodnotu, proměnná *ConW* , nastavuje algoritmus sám podle hodnoty *gW*. Záleží totiž na hodnotě *tiW=0*, která může být časovou konstantou pro filtr prvního řádu nebo lineární interpolací žádané hodnoty v násobcích cyklu automatu. Tohoto parametru jsem nevyužil. Odchylka *Dev* skutečné hodnoty od žádané je vypočítávána procentuelně na rozsahu měřené veličiny. Proměnná *Output* představuje výstup žádaný algoritmem, *LastOut* je minulý akční zásah. *CurOut* a *ConOut* je výstup skutečně žádaný v daném kroku a výstup regulátorem realizovaný. *MinU=0* a *MaxU=10000* je minimální a maximální povolený akční zásah. Maximální povolený přírůstek akčního zásahu je *dMaxU*. Délka výstupního cyklu *OutCycle=4* je perioda vzorkování v setinách sekundy. Tato hodnota představuje periodu vzorkování 40ms. Takto nízkou hodnotu jsem mohl nastavit, neboť celý uživatelský program se skládal jen z několika pomocných instrukcí pro čtení a zápis dat a instrukce PID regulátoru a tudíž doba jednoho cyklu programu byla velmi malá. Nastavení regulátoru se provádí naplněním bitů řídícího slova *Control=68*. Regulátor se může nacházet v režimu automatickém, ručním nebo havarijním. Může pracovat jako regulátor s přímým nebo přírůstkovým algoritmem. Dále je možné nastavit kaskádní nebo časově proporcionální řízení. V mém případě se jedná o přímé řízení v automatickém režimu s unifikovaným výstupem normovaným na rozsah 12-ti bitového výstupního převodníku. Vlastní parametry seřízení regulátoru vycházejí z jeho průmyslového využití. Pásmo proporcionality *Pbnd=2000* určuje zesílení vztahem *K=1000/Pbnd*. Je určeno rozsahem vstupních a výstupních hodnot. Ve svém důsledku potom znamená kolik procent rozsahu vstupního signálu způsobí právě změnu v celém rozsahu signálu výstupního. *RelCool=1000* je pomocné pásmo

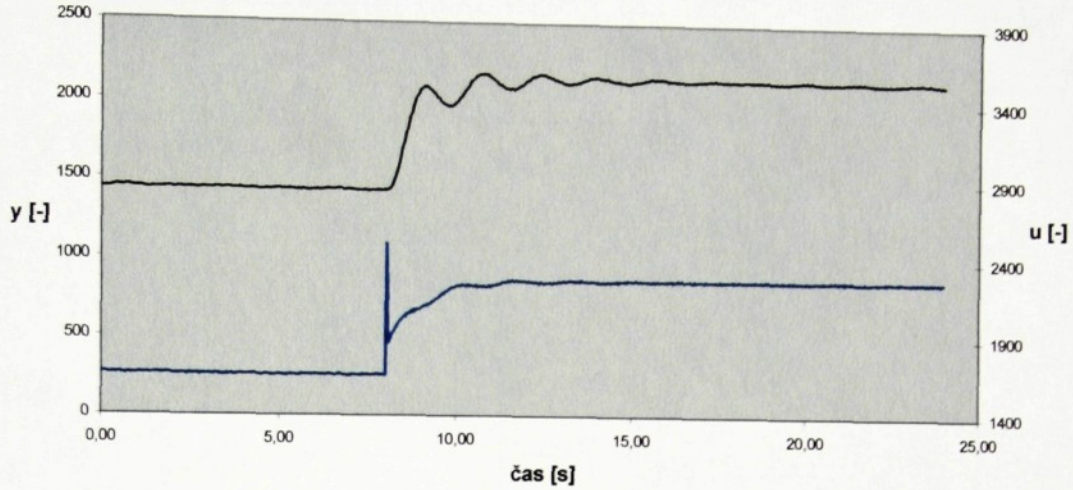
proporcionality pro zápornou odchylku. V tomto případě je tato složka bez vlivu. *Ti=2* je integrační konstanta v desetinách sekundy. Pro nulovou hodnotu je integrační složka vypnutá. *Td=5* je derivační konstanta (také desetiny s). Symetrické pásmo necitlivosti *Egap=0* je nevyužito. Symetrické pásmo odchylky, ve kterém působí derivační složka *Dgap=10000*. To znamená, že derivační složka působí stále. *Igap=10000* je symetrické pásmo odchylky, ve kterém působí integrační složka (působí také stále).

Program RegPid1.jlr představuje naprogramovaný algoritmus regulátoru pomocí instrukce PID. Funkci tohoto programu jsem testoval na PLC TC606 a poté jsem provedl experimentální seřízení regulátoru na uvažované regulované soustavě s následujícími výsledky.

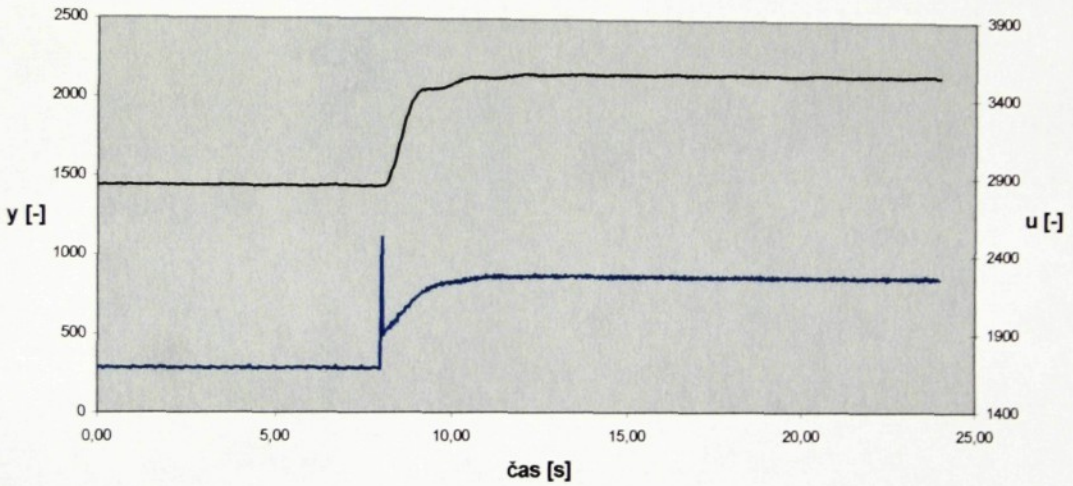
Na obr.5.42 je odezva výstupní veličiny na skokovou změnu žádané hodnoty při seřízení *Pbnd=2000*, *Ti=5* a *Td=0*. Na obr.5.43 je odezva výstupní veličiny na skokovou změnu žádané hodnoty při seřízení *Pbnd=2000*, *Ti=2* a *Td=2* a na obr.5.44 při seřízení *Pbnd=2000*, *Ti=2* a *Td=5*.



Obr. 5.42 Odezva výstupní veličiny regulované soustavy



Obr. 5.43 Odezva výstupní veličiny regulované soustavy



Obr. 5.44 Odezva výstupní veličiny regulované soustavy

Uvedené experimentální seřízení PID regulátoru je provedeno při vzorkovací periodě 40ms. Pro vyšší hodnoty vzorkovací periody například 100ms byly naměřené regulační pochody značně horší a seřízení regulátoru muselo být provedeno na velmi pomalý náběh regulované veličiny.

Při porovnání naměřených regulačních pochodů obou regulátorů a na základě experimentů a zkušeností s nasazením v laboratoři se ukázalo, že použitím navrženého adaptivního regulátoru lze dosáhnout lepších výsledků.

6. Závěr

V předkládané práci je vypracován projekt adaptivní regulace. Jedná se o adaptivní regulátor realizovaný PLC automatem TECOMAT, jenž je produktem společnosti TECO a.s. Vývojovým prostředím, ve kterém jsem vytvořil program adaptivního regulátoru, byl programový produkt určený k tvorbě aplikací pro PLC TECOMAT, Epos for Windows. Nedílnou součástí této práce je navržená vizualizační aplikace v prostředí pro tvorbu vizualizací aplikací řídicích systémů TECOMAT Reliance, která slouží jako vyšší uživatelské rozhraní. Pro základní interakci s uživatelem je naprogramován operační panel PLC.

Ověření funkce a vlastností tohoto adaptivního regulátoru bylo provedeno v laboratoři KŘT na dvou různých dynamických soustavách. První je relativně rychlou a kmitavou soustavou. Jedná se o regulaci otáček tachodynama spojeného pružným hřídelem se stejnosměrným elektromotorem. Druhou je pomalá tepelná soustava, která je představována průtokovým ohřívačem vody. Podrobný rozbor výsledků regulace první soustavy je i předmětem této práce. Dále jsem porovnával dosažené výsledky regulace s řízením uvažované soustavy pomocí PID regulátoru, který je součástí programového vybavení řídicího systému TECOMAT v podobě systémové instrukce. Na základě experimentů a zkušeností s nasazením v laboratoři se ukazuje, že vlastní regulační pochody jsou lepší než při použití standardního PID regulátoru řídicího systému TECOMAT.

Hlavní výhody tohoto projektu adaptivní regulace spatřuji v následujících skutečnostech:

a) Přímo v provozu je možné provést identifikaci regulované soustavy a získat model soustavy reprezentovaný diskrétním Z přenosem.

b) Program provádí automatické seřízení číslicového regulátoru na optimální parametry.

c) Pomalé změny parametrů regulované soustavy lze postihnout průběžnou identifikací, kdy v každém diskrétním časovém kroku je vypočítáván diskrétní model a probíhá návrh parametrů regulátoru. Nebo

způsobem, kdy identifikace je spuštěna jednou za čas, jen na určitou dobu. Ukázalo se, že optimálních výsledků lze dosáhnout právě tímto způsobem.

d) Systém umožní dosáhnout snížení vzorkovací periody za předpokladu přepočtu získaného diskrétního modelu a navržení optimálních parametrů regulátoru a následnou implementaci takto získaného číslicového regulátoru.

Závěrem se dá konstatovat, že se podařilo implementovat zvolené algoritmy adaptivní regulace na tento řídicí systém a v této podobě je tedy adaptivní regulátor připraven k průmyslovému ověření ve spolupráci s firmou TECO a.s.

7. Seznam použité literatury

[1] Peterka,V. a kol.: Algoritmy pro adaptivní mikroprocesorovou regulaci technologických procesů. Praha, ÚTIA, ČSAV 1982.

[2] Bobál,V.-Böhm,J.-Prokop,R.-Fessl,J.: Praktické aspekty samočinně se nastavujících regulátorů. VUT Brno, VUTIUM 1999.

[3] Hanuš,B.-Olehla,M.-Modrlák,O.: Číslicová regulace technologických procesů. VUT Brno, VUTIUM 2000.

[4] Janeček,B.: Metoda dynamického programování použitá pro číslicovou regulaci lineárních dynamických soustav. Habilitační práce, TU Liberec, 1996, str.6-36.

[5] Modrlák,O.: Adaptivní řízení. Skripta VŠST Liberec 1987.

[6] Modrlák,O.: Teorie automatického řízeni II. Skripta VŠST Liberec 1992.

[7] TECO a.s.: Příručka programátora PLC TECOMAT, Uživatelský manuál TXV 001 09.01, 2000.

[8] Kubec,J.: Příručka pro projektování systémů TECOMAT, Uživatelský manuál TXV 001 08.01, 1997.

[9] TECO a.s.: Technické vybavení programovatelných automatů řady TC500, Uživatelský manuál TXV 138 07.01, 1998.

[10] TECO a.s.: Modulární a kompaktní programovatelné automaty TECOMAT, Uživatelský manuál TXV 001 99.01, 1998.

[11] TECO a.s.: Soubor instrukcí PLC TECOMAT, Uživatelský manuál TXV 001 05.01, 1999.

[12] Cach,P.: Průvodce programem Epos For Windows, Uživatelský manuál TXV 138 57.01, 1999.

[13] TECO a.s.: Programovací jazyk JLR, Uživatelský manuál TXV 138 54.01, 1999.

[14] TECO a.s.: Analyzátor pro systémy TECOMAT a TECOREG, Uživatelský manuál TXV 138 53.01,1998.

[15] Zajíček,R.: Návod k používání programu PanelTool, Uživatelský manuál TXV 138 52.01, 1997.

[16] TECO a.s.: Manuál Reliance, Uživatelský manuál TXV 003 02.01, 1999

[17] TECO a.s.: WWW stránky firmy TECO a.s.

8. Seznam příloh

Příloha č.1 : Výpis naprogramovaného algoritmu průběžného odhadování parametrů metodou LD rozkladu.

Příloha č.2 : Výpis naprogramovaného algoritmu metody dynamického programování pro návrh číslicového regulátoru.

Příloha č.3 : Zdrojový program adaptivního regulátoru pro PLC TC606.

Příloha č.4 : Přiložená disketa se zdrojovými programy.

Příloha č.1:

```
%program ldfrun.m je naprogramovaný algoritmus průběžného odhadování
%parametrů metodou LD rozkladu

    HU=DA(NDA1);
    W=DU(NDA1)*HU;
    G(NDA1)=W;
    SA=FR+W*HU;
    DU(NDA1)=DU(NDA1)/SA;
    SI=SA;
    I1=2;
    K=NVU1;
    J=NDA2;
    while J>0
       HU=DA(J);
       J1=J+1;
       K1=K;
       for I=J1:NDA1
          HU=HU+VU(K1)*DA(I);
          K1=K1+1;
       end
       AU=HU/SI;
       W1=DU(J);
       W=W1*HU;
       G(J)=W;
       SA=SI+W*HU;
       DU(J)=SI/FR*W1/SA;
       SI=SA;
       K1=K;
       for I=J1:NDA1
          W=VU(K1);
          VU(K1)=W-G(I)*AU;
          G(I)=G(I)+W*G(J);
          K1=K1+1;
       end
       K=K-I1;
       I1=I1+1;
       J=J-1;
    end
```

Příloha č.2:

```
%program dynrun.m je naprogramovaný algoritmus metody dynamického
%programování pro návrh číslicového regulátoru

  for I=1:NA2
     GG(I)=PolA(I)-PolA(I+1);
  end
  GG(NA1)=PolA(NA1);
  for I=1:NA1
     if I<=NB1
        GG(I+NA-1)=-PolB(I);
     else
        GG(I+NA-1)=0.0;
     end
  end
  %%
  I4=NA; I5=3;
  for K=1:2
     K1=N3A;
     for I=1:NA1
        I1=I+1; I2=I4; I3=K1; J1=NA; W=0.0;
        for J=1:I
           I3=I3+J1-1; W=W+GG(I2)*PP(I3); J1=J1-1; I2=I2+1;
        end;
        for J=I1:NA1
           I3=I3+1; W=W+GG(I2)*PP(I3); I2=I2+1;
        end;
        PP(I5)=W; I5=I5+1; K1=K1+1;
     end;
     I4=1; I5=N3A;
  end;
  %%
  I1=1; I2=NA; I3=3;
  for I=1:2
     for J=I:2
        W=0.0; I4=I3; I5=I2;
        for K=1:NA1
           W=W+GG(I5)*PP(I4); I4=I4+1; I5=I5+1;
        end;
        PP(I1)=W; I1=I1+1; I3=N3A;
     end;
     I1=N2A; I2=1;
  end;
  %%
  PP(1)=PP(1)+Kappa; PP(N2A)=PP(N2A)+1.0; W=PP(1);
  I1=N1PP; I2=N3PP;
  I=NA;
  while I>1
     J=NA;
     while J>=I
        PP(I1)=PP(I2)-PP(I)/W*PP(J); I1=I1-1; I2=I2-1; J=J-1;
     end;
     I2=I2-1; I=I-1;
  end;
  %%
  for I=1:NA1
     StateReg(I)=PP(I+1)/W;
  end
```

```
%%
SumCoePolN=StateReg(1); PolN(1)=SumCoePolN;
for J=2:NA1
   I1=2; W=0.0;
   for I=J:NA1
      W=W+StateReg(I1)*GG(I); I1=I1+1;
   end;
   SumCoePolN=SumCoePolN+W; PolN(J)=W;
end;
%%
for J=2:NA1
   I1=2; W=0.0;
   for I=J:NA1
      W=W+StateReg(I1)*GG(I+NA-1); I1=I1+1;
   end;
   PolM(J)=W;
end;
```

Příloha č.3:

```
Program adapt12bit; {v07_400ms_4RAD_10V_12BIT_LOG1}

{program pracuje se vzorkovací periodou 400ms, diskrétní model soustavy je 4-tého řádu}
{vstupní AD převodník je 12-ti bitový s rozsahem 0-10V, výstupní DA převodník je 12-ti}
{bitový s rozsahem 0-10V, napěťové úrovně výstupní veličiny "y" a akční veličiny "u"}
{jsou reprezentovány číslem typu Word s rozsahem 0-4095}

{deklarační část programu - deklarece konstant, typů a proměnných}
const
   FR          : byte = 1;{koeficient zapomínání}
   NA1         : byte = 4;{řád modelu}
   NB1         : byte = 3;
   NDA2        : byte = (NA1+NB1)+1;
   NDA1        : byte = (NDA2)+1;
   NVU1        : byte = ((NDA1*(NDA1+1))/2)-1;
   NABS1MAX    : byte = NA1;
   T           : byte = 100;
   Tdelay      : byte = T/10;

   NA2         : byte = (NA1)-1;
   NA          : byte = (NA1)+1;
   N1A         : byte = (NA)+1;
   N2A         : byte = (N1A)+1;
   N3A         : byte = (N2A)+1;
   NG1         : byte = (2*NA1)+1;
   NPP         : byte = (2*N1A)+1;
   N1PP        : byte = ((N2A*N3A)/2)-1;
   N3PP        : byte = (N1PP)-2;
   UMax        : float = 4095;
   UMin        : float = 0;
type
   ArDA        = array[0..NDA1] of real;
   ArVU        = array[0..NVU1] of real;
   ArABS       = array[0..NABS1MAX] of real;

   ArGG        = array[0..NG1] of real;
   ArPP        = array[0..N1PP] of real;
var
   DA,DU,G,DAM       : ArDA;{vektor měření DA}
   VU                : ArVU;{vektor parametrů VU}
   PolA,PolB         : ArABS;{jmenovatel a čitatel přenosu diskrétního modelu}
   Y,U,UOut,UOld     : float;{výstupní a akční veličina y, u}
   YW,UW             : word;{pracovní proměnné programu}
   YSteady,USteady   : float;{počáteční hodnoty y a u v pracovním bodě}
   YModel,WReg       : float;{simulovaná odezva modelu ym, žádaná hodnota w}
   YDisc,DUDisc      : ArABS;{diskrétní časové řady výstupní a akční veličiny y, u}
   VstupW,VstupM     : word;{vzorkované hodnoty regulované veličiny}
   Vstup  at %X0     : word;{obraz analogového vstupu AI0}
   Vystup at %Y0     : word;{obraz analogového výstupu AO0}
   Count             : word;{pracovní proměnná programu}
   Start,Period      : bool;{řídící proměnné programu}
   SA,SI,HU,AU,W,W1  : float;{pracovní proměnné programu}
   NSteady,NModel    : byte;{pomocné proměnné programu}
   NIdent,J          : byte;{pomocné proměnné programu}
   CmInit,CmSample   : bool;{řídící proměnné programu}
   CmSteady,CmIdent  : bool;{řídící proměnné programu}
   CmDyn,CmReg       : bool;{řídící proměnné programu}

   GG                : ArGG;{pracovní proměnné programu}
   PP                : ArPP;{pracovní proměnné programu}
   PolM,PolN         : ArABS;{jmenovatel a čitatel přenosu číslicového regulátoru}
   PolW,StateReg     : ArABS;{stavový regulátor}
   SumCoePolN        : float;{součet koeficentů čitatele přenosu číslicového regulátoru}
   Kappa             : float;{koeficient tlumení regulace}
   FlSteady,FlIdent  : bool;{proměnné příznaků}
   FlDyn             : bool;{proměnné příznaků}
   AOUT_12B          : word;{pomocná proměnná pro zápis na 12-ti bitový převodník}

{deklarace uživatelské instrukce USI pro zápis na výstupní 12-ti bitový DA převodník}
usis
  OT13_12B = Ot13_12b.uid;
```

```
{deklarace tabulky popisující napětové rozsahy vstupního 12-ti bitového AD převodníku}
tables
T1:table[1] of byte=00000000;
{deklarace vstupních a výstupních jednotek PLC TC606 - adresa, typ, počet, obraz v
zápisníku}
units
unit 0,0,$D0,2,0,%X0,On,@T1;

{vložený soubor s deklarací pro operační panel}
//include identf1.ID0;

begin
{$T+}
{základní proces P0}
  "P 0;
  "E 0;
{proces jednou za 400ms P5}
  "P 5;
  {vložený soubor pro zpracovaní operačního panelu}
  //include identf1.I00;

      {command init - inicializace proměnných}
      if CmInit then
        begin
          for I:=0 to NDA1 do DU[I]:=1.0E5;
          for I:=0 to NVU1 do VU[I]:=0.0;
          for I:=0 to NDA2 do DA[I]:=0.0;
          DA[NDA1]:=1.0;
          for I:=0 to NABS1MAX do PolA[I]:=0.0;
          for I:=0 to NABS1MAX do PolB[I]:=0.0;
          PolA[0]:=1.0;
          YSteady:=0.0; USteady:=0.0; YModel:=0.0;
          NSteady:=0; NModel:=0; NIdent:=0;
          for I:=0 to N1PP do PP[I]:=0.0;
          PP[NPP]:=1.0;
          PolN:=PolA;
          PolM:=PolA;
          Kappa:=5; WReg:=0.0;
          FlSteady:=false; FlIdent:=false; FlDyn:=false;
        end;

      {command reg  - regulace, výpočet akčního zásahu}
      if CmReg then
        begin
          Y:=Vstup;
          YW:=Y;
          W:=PolN[0]*Y-SumCoePolN*WReg;
          for I:=1 to NA1 do W:=W+PolN[I]*YDisc[I]-PolM[I]*DUDisc[I];
          U:=U+W;                                          {výpočet akčního zásahu w}
          if [U > UMax] then U:=UMax;
          if [U < UMin] then U:=UMin;
          Vystup:=U;
          UW:=U;
          DUDisc[0]:=UW-UOut;
          UOut:=UW;
        end
      else
        if CmSample then
          begin
            WReg:=Vstup;
            if [U > UMax] then U:=UMax;
            if [U < UMin] then U:=UMin;
            Vystup:=U;
            UW:=U;
            UOut:=UW;
          end;

      {command sample - vzorkování, příprava dat}
      if CmSample then
        begin
          CmInit:=false;
          Y:=Vstup;
          VstupW:=Y;
          for I:=0 to NA1-1 do DA[NA1-I]:=DA[NA1-I-1];          {posun DA část y}
```

```
    for I:=0 to NB1-1 do DA[NB1-I+NA1+1]:=DA[NB1-I+NA1+1-1]; {posun DA část u}
    if CmIdent then
      begin
        DA[0]:=Y-YSteady;                                    {napnění DA hodnotou yk}
        DA[NA1+1]:=UOld-USteady;                             {napnění DA hodnotou uk-1}
      end
    else
      begin
        DA[0]:=Y;                                            {napnění DA hodnotou yk}
        DA[NA1+1]:=UOld;                                     {napnění DA hodnotou uk-1}
      end;
    UOld:=UOut;                                              {zapamatování hodnoty uk}
    YDisc[0]:=Y;
    for I:=0 to NA1-1 do YDisc[NA1-I]:=YDisc[NA1-I-1];
    for I:=0 to NA1-1 do DUDisc[NA1-I]:=DUDisc[NA1-I-1];
  end
else CmInit:= not false;

{command steady - výpočet střední hodnoty "y" v počátečním pracovním bodě}
if CmSteady then
  begin
    CmIdent := false;
    if !NSteady < 64! then
      begin
        inc(NSteady);
        YSteady:=(NSteady-1)/NSteady*YSteady+Y/NSteady;
      end
    else
      begin
        CmSteady := false;
        NSteady := 0;
        FlSteady:= not false;
      end;
    USteady:=UOut;
  end;

{command ident - algoritmus LD-filtru pro identifikaci dynamické soustavy}
if CmIdent then
  begin
  if !NIdent < 8! then inc(NIdent)
  else
    begin
    HU:=DA[NDA1];
    W:=DU[NDA1]*HU;
    G[NDA1]:=W;
    SA:=FR+W*HU;
    DU[NDA1]:=DU[NDA1]/SA;
    SI:=SA;

      HU:=DA[NDA2-0];

            HU:=HU+VU[NVU1]*DA[NDA1];

      AU:=HU/SI;
      W1:=DU[NDA2-0];
      W:=W1*HU;
      G[NDA2-0]:=W;
      SA:=SI+W*HU;
      DU[NDA2-0]:=SI/FR*W1/SA;
      SI:=SA;

            W:=VU[NVU1]; VU[NVU1]:=W-G[NDA1]*AU; G[NDA1]:=G[NDA1]+W*G[NDA2-0];


      HU:=DA[NDA2-1];
        for I:=0 to 1 do
          begin
            HU:=HU+VU[NVU1-2+I]*DA[NDA1-1+I];
          end;
      AU:=HU/SI;
      W1:=DU[NDA2-1];
      W:=W1*HU;
      G[NDA2-1]:=W;
      SA:=SI+W*HU;
```

```
            DU[NDA2-1]:=SI/FR*W1/SA;
            SI:=SA;
              for I:=0 to 1 do
                begin
                  W:=VU[NVU1-2+I]; VU[NVU1-2+I]:=W-G[NDA1-1+I]*AU; G[NDA1-1+I]:=G[NDA1-
1+I]+W*G[NDA2-1];
                end;

            HU:=DA[NDA2-2];
              for I:=0 to 2 do
                begin
                  HU:=HU+VU[NVU1-5+I]*DA[NDA1-2+I];
                end;
            AU:=HU/SI;
            W1:=DU[NDA2-2];
            W:=W1*HU;
            G[NDA2-2]:=W;
            SA:=SI+W*HU;
            DU[NDA2-2]:=SI/FR*W1/SA;
            SI:=SA;
              for I:=0 to 2 do
                begin
                  W:=VU[NVU1-5+I]; VU[NVU1-5+I]:=W-G[NDA1-2+I]*AU; G[NDA1-2+I]:=G[NDA1-
2+I]+W*G[NDA2-2];
                end;

            HU:=DA[NDA2-3];
              for I:=0 to 3 do
                begin
                  HU:=HU+VU[NVU1-9+I]*DA[NDA1-3+I];
                end;
            AU:=HU/SI;
            W1:=DU[NDA2-3];
            W:=W1*HU;
            G[NDA2-3]:=W;
            SA:=SI+W*HU;
            DU[NDA2-3]:=SI/FR*W1/SA;
            SI:=SA;
              for I:=0 to 3 do
                begin
                  W:=VU[NVU1-9+I]; VU[NVU1-9+I]:=W-G[NDA1-3+I]*AU; G[NDA1-3+I]:=G[NDA1-
3+I]+W*G[NDA2-3];
                end;

            HU:=DA[NDA2-4];
              for I:=0 to 4 do
                begin
                  HU:=HU+VU[NVU1-14+I]*DA[NDA1-4+I];
                end;
            AU:=HU/SI;
            W1:=DU[NDA2-4];
            W:=W1*HU;
            G[NDA2-4]:=W;
            SA:=SI+W*HU;
            DU[NDA2-4]:=SI/FR*W1/SA;
            SI:=SA;
              for I:=0 to 4 do
                begin
                  W:=VU[NVU1-14+I]; VU[NVU1-14+I]:=W-G[NDA1-4+I]*AU; G[NDA1-
4+I]:=G[NDA1-4+I]+W*G[NDA2-4];
                end;

            HU:=DA[NDA2-5];
              for I:=0 to 5 do
                begin
                  HU:=HU+VU[NVU1-20+I]*DA[NDA1-5+I];
                end;
            AU:=HU/SI;
            W1:=DU[NDA2-5];
            W:=W1*HU;
            G[NDA2-5]:=W;
            SA:=SI+W*HU;
            DU[NDA2-5]:=SI/FR*W1/SA;
            SI:=SA;
              for I:=0 to 5 do
                begin
                  W:=VU[NVU1-20+I]; VU[NVU1-20+I]:=W-G[NDA1-5+I]*AU; G[NDA1-
5+I]:=G[NDA1-5+I]+W*G[NDA2-5];
                end;
```

```
            HU:=DA[NDA2-6];
              for I:=0 to 6 do
                begin
                  HU:=HU+VU[NVU1-27+I]*DA[NDA1-6+I];
                end;
            AU:=HU/SI;
            W1:=DU[NDA2-6];
            W:=W1*HU;
            G[NDA2-6]:=W;
            SA:=SI+W*HU;
            DU[NDA2-6]:=SI/FR*W1/SA;
            SI:=SA;
              for I:=0 to 6 do
                begin
                  W:=VU[NVU1-27+I]; VU[NVU1-27+I]:=W-G[NDA1-6+I]*AU; G[NDA1-
6+I]:=G[NDA1-6+I]+W*G[NDA2-6];
                end;

            HU:=DA[NDA2-7];
              for I:=0 to 7 do
                begin
                  HU:=HU+VU[NVU1-35+I]*DA[NDA1-7+I];
                end;
            AU:=HU/SI;
            W1:=DU[NDA2-7];
            W:=W1*HU;
            G[NDA2-7]:=W;
            SA:=SI+W*HU;
            DU[NDA2-7]:=SI/FR*W1/SA;
            SI:=SA;
              for I:=0 to 7 do
                begin
                  W:=VU[NVU1-35+I]; VU[NVU1-35+I]:=W-G[NDA1-7+I]*AU; G[NDA1-
7+I]:=G[NDA1-7+I]+W*G[NDA2-7];
                end;

            HU:=DA[NDA2-8];
              for I:=0 to 8 do
                begin
                  HU:=HU+VU[NVU1-44+I]*DA[NDA1-8+I];
                end;
            AU:=HU/SI;
            W1:=DU[NDA2-8];
            W:=W1*HU;
            G[NDA2-8]:=W;
            SA:=SI+W*HU;
            DU[NDA2-8]:=SI/FR*W1/SA;
            SI:=SA;
              for I:=0 to 8 do
                begin
                  W:=VU[NVU1-44+I]; VU[NVU1-44+I]:=W-G[NDA1-8+I]*AU; G[NDA1-
8+I]:=G[NDA1-8+I]+W*G[NDA2-8];
                end;

          for I:=1 to NA1 do PolA[I]:=VU[I-1];            {diskrétní přenos modelu B,A}
          for I:=0 to NB1 do PolB[I]:=-VU[I+NA1];
          if !NModel < 84! then
            begin
              inc(NModel);
              if !NModel = 84! then DAM:=DA;
            end
          else                                          {výpočet simulace odezvy modelu}
            begin
              for I:=0 to NA1-1 do DAM[NA1-I]:=DAM[NA1-I-1];
              for I:=0 to NB1-1 do DAM[NB1-I+NA1+1]:=DAM[NB1-I+NA1+1-1];
              DAM[NA1+1]:=DA[NA1+1];
              YModel:=0.0;
              for I:=1 to NA1 do YModel:=YModel-PolA[I]*DAM[I];
              for I:=0 to NB1 do YModel:=YModel+PolB[I]*DAM[I+NA1+1];
              DAM[0]:=YModel;
              YModel:=YModel+YSteady;
              VstupM:=YModel;
            end;
          FlIdent:= not false;
          end;
        end
      else
        begin
          NIdent:=0;
          NModel:=0;
```

```
    YModel:=0.0;
  end;

{command dyn - algoritmus DYN-prog pro návrh optimálních parametrů regulátoru}
if CmDyn then
  begin
    J:=0;
    for I:=0 to NA2 do GG[I]:=PolA[I]-PolA[I+1];
    GG[NA1]:=PolA[NA1];
    for I:=0 to NA1 do
      begin
        if !J<=NB1! then GG[I+NA]:=-PolB[I] else GG[I+NA]:=0.0;
        inc(J);
      end;
    {}
    W:=0.0;
    for I:=1 to 5 do
      begin
        W:=W+GG[I+4]*PP[I+12];
      end;
    PP[2]:=W;
    W:=0.0;
    W:=W+GG[5]*PP[14];
    for I:=1 to 4 do
      begin
        W:=W+GG[I+5]*PP[I+17];
      end;
    PP[3]:=W;
    W:=0.0;
    W:=W+GG[5]*PP[15];
    W:=W+GG[6]*PP[19];
    for I:=1 to 3 do
      begin
        W:=W+GG[I+6]*PP[I+21];
      end;
    PP[4]:=W;
    W:=0.0;
    W:=W+GG[5]*PP[16];
    W:=W+GG[6]*PP[20];
    W:=W+GG[7]*PP[23];
    for I:=1 to 2 do
      begin
        W:=W+GG[I+7]*PP[I+24];
      end;
    PP[5]:=W;
    W:=0.0;
    W:=W+GG[5]*PP[17];
    W:=W+GG[6]*PP[21];
    W:=W+GG[7]*PP[24];
    for I:=1 to 2 do
      begin
        W:=W+GG[I+7]*PP[I+25];
      end;
    PP[6]:=W;

    W:=0.0;
    for I:=1 to 5 do
      begin
        W:=W+GG[I-1]*PP[I+12];
      end;
    PP[8]:=W;
    W:=0.0;
    W:=W+GG[0]*PP[14];
    for I:=1 to 4 do
      begin
        W:=W+GG[I]*PP[I+17];
      end;
    PP[9]:=W;
    W:=0.0;
    W:=W+GG[0]*PP[15];
    W:=W+GG[1]*PP[19];
    for I:=1 to 3 do
      begin
        W:=W+GG[I+1]*PP[I+21];
      end;
    PP[10]:=W;
    W:=0.0;
    W:=W+GG[0]*PP[16];
    W:=W+GG[1]*PP[20];
    W:=W+GG[2]*PP[23];
```

```
    for I:=1 to 2 do
      begin
        W:=W+GG[I+2]*PP[I+24];
      end;
    PP[11]:=W;
    W:=0.0;
    W:=W+GG[0]*PP[17];
    W:=W+GG[1]*PP[21];
    W:=W+GG[2]*PP[24];
    for I:=1 to 2 do
      begin
        W:=W+GG[I+2]*PP[I+25];
      end;
    PP[12]:=W;
    {}
    W:=0.0;
    for I:=1 to 5 do
      begin
        W:=W+GG[I+4]*PP[I+1];
      end;
    PP[0]:=W;
    W:=0.0;
    for I:=1 to 5 do
      begin
        W:=W+GG[I+4]*PP[I+7];
      end;
    PP[1]:=W;

    W:=0.0;
    for I:=1 to 5 do
      begin
        W:=W+GG[I-1]*PP[I+7];
      end;
    PP[7]:=W;
    {}
    PP[0]:=PP[0]+Kappa;
    PP[N2A]:=PP[N2A]+1.0;
    W:=PP[0];

  PP[27]:=PP[25]-PP[5]/W*PP[5];

  for I:=1 to 2 do
    begin
      PP[27-I]:=PP[24-I]-PP[4]/W*PP[6-I];
    end;
  for I:=1 to 3 do
    begin
      PP[25-I]:=PP[21-I]-PP[3]/W*PP[6-I];
    end;
  for I:=1 to 4 do
    begin
      PP[22-I]:=PP[17-I]-PP[2]/W*PP[6-I];
    end;
  for I:=1 to 5 do
    begin
      PP[18-I]:=PP[12-I]-PP[1]/W*PP[6-I];
    end;
  {}
  for I:=0 to NA1 do StateReg[I]:=PP[I+1]/W;                    {stavový regulátor}
  {}
  SumCoePolN:=StateReg[0];
  PolN[0]:=SumCoePolN;
  W:=0.0;
  for I:=1 to 4 do
    begin
      W:=W+StateReg[I]*GG[I];
    end;
  SumCoePolN:=SumCoePolN+W;
  PolN[1]:=W;
  W:=0.0;
  for I:=1 to 3 do
    begin
      W:=W+StateReg[I]*GG[I+1];
    end;
  SumCoePolN:=SumCoePolN+W;
  PolN[2]:=W;
  W:=0.0;
  for I:=1 to 2 do
    begin
      W:=W+StateReg[I]*GG[I+2];
```

```
          end;
        SumCoePolN:=SumCoePolN+W;
        PolN[3]:=W;
        W:=0.0;

            W:=W+StateReg[1]*GG[1+3];

        SumCoePolN:=SumCoePolN+W;
        PolN[4]:=W;
        {}
        W:=0.0;
        for I:=1 to 4 do
          begin
            W:=W+StateReg[I]*GG[I+NA];
          end;
        PolM[1]:=W;
        W:=0.0;
        for I:=1 to 3 do
          begin
            W:=W+StateReg[I]*GG[I+1+NA];
          end;
        PolM[2]:=W;
        W:=0.0;
        for I:=1 to 2 do
          begin
            W:=W+StateReg[I]*GG[I+2+NA];
          end;
        PolM[3]:=W;
        W:=0.0;

            W:=W+StateReg[1]*GG[1+3+NA];

        PolM[4]:=W;
        {}                                          {diskrétní přenos regulátoru N,M}

        FlDyn:= not false;
      end;
  "E 5;

{proces P62 teplý restart}
  "P 62;
  {vložený soubor pro nastavení operačního panelu}
  //include identf1.I63;
  "E 62;

{proces P63 studený restart}
  "P 63;
  {vložený soubor pro nastavení operačního panelu}
  //include identf1.I63;
  "E 63;

{ukončující proces P64}
  "P 64;
       AOUT_12B:=Vystup;       {zápis pomocí mnemonických instrukcí - adresa proměnné}
       "ld     @.AOUT_12B;     {od začátku registru}
       setusi(OT13_12B);       {zápis USI instrukce pro obsluhu 12-ti bitového DA
převodníku}
  "E 64;
end.
```