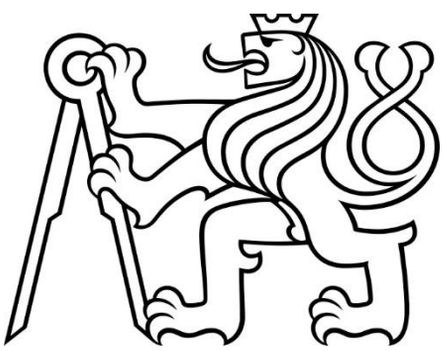
ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

FAKULTA
STROJNÍ

ÚSTAV PŘÍSTROJOVÉ
A ŘÍDICÍ TECHNIKY



RELÉOVÁ IDENTIFIKACE PRO PLC TECOMAT FOXTROT

DIPLOMOVÁ
PRÁCE

2018

ALŽBĚTA
HORNYCHOVÁ

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hornychová** Jméno: **Alžběta** Osobní číslo: **420473**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav přístrojové a řídící techniky**

Studijní program: **Strojní inženýrství**

Studijní obor: **Přístrojová a řídicí technika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Reléová identifikace pro PLC Tecomat Foxtrot

Název diplomové práce anglicky:

Relay identification for PLC Tecomat Foxtrot

Pokyny pro vypracování:

1. Seznamte se s vybranými algoritmy reléové zpětnovazební identifikace pro odhad modelu řízené soustavy typu SISO.
2. Seznamte se s HW a SW prostředky systému Tecomat Foxtrot firmy Teco a.s.
3. Naprogramujte vybrané algoritmy reléové zpětnovazební identifikace použitelné pro automatické naladění PID regulátoru.
4. Experimentálně ověřte a porovnejte funkčnost vytvořených programových modulů reléové identifikace.
5. Vybraný programový modul reléové identifikace upravte do tvaru vhodného pro implementaci do knihovny iControlLib pro prostředí Mosaic.

Seznam doporučené literatury:

[1] Chidambaram M. and Sathe V. (2014). Relay Autotuning for Identification and Control. Cambridge University Press, Cambridge

[2] Äström K. J. and Hägglund T. (2006). Advanced PID Control. ISA - The Instrumentation, Systems and Automation Society

[3] Berner, J., Hägglund, T., Äström K.J. (2016): Improved relay autotuning using normalized time delay, American Control Conference (ACC), Boston, pp. 1869-1875

[4] Hofreiter M.(2017). Biased-Relay Feedback Identification for Time Delay Systems. IFAC-PapersOnLine, Elsevier, Volume 50, Issue 1, pp. 14620-14625

Jméno a pracoviště vedoucí(ho) diplomové práce:

prof. Ing. Milan Hofreiter, CSc., U12110.3

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **18.04.2018** Termín odevzdání diplomové práce: **15.06.2018**

Platnost zadání diplomové práce: _____

prof. Ing. Milan Hofreiter, CSc. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Michael Valášek, DrSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomantka bere na vědomí, že je povinna vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ Datum převzetí zadání

_____ Podpis studentky

Poděkování

Mé poděkování patří v prvé řadě vedoucímu práce profesoru Milanovi Hofreiterovi za konzultace a vedení při vytváření této práce. Dále pak patří inženýru Milanovi Bydžovskému za poskytnutí vzorového programu pro PLC, Františkovi Hylmarovi za pomoc při hledání těžko odhalitelných syntaktických chyb a inženýrům Petrovi Válovi, Danielovi Makovičkovi a Liborovi Válovi za zasvěcení do základních programátorských postupů.

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně s tím, že její výsledky mohou být dále použity podle uvážení vedoucího diplomové práce jako jejího spoluautora. Souhlasím také s případnou publikací výsledků diplomové práce nebo její podstatné části, pokud budu uvedena jako její spoluautor.

V Praze, 15. června 2018

Abstrakt

V rámci této práce byly pro PLC Tecomat Foxtrot napsány skripty reléové zpětnovazební identifikace podle tří různých metod. Při volbě metod byl kladen důraz na jednoduchost měření a následných výpočtů. První z použitých metod byla publikována v práci Biased-Relay Feedback Identification for Time Delay Systems (Hofreiter, 2017). Tato metoda dokázala nalézt body Nyquistovy frekvenční charakteristiky, avšak při určování parametrů modelu druhého řádu s dopravním zpožděním docházelo k případům, kdy určené dopravní zpoždění bylo záporné a mělo tedy nesmyslnou hodnotu. Druhá metoda je upravenou verzí metody první a je popsána v článku Alternative Identification Method using Biased Relay Feedback (Hofreiter, 2018). Touto metodou se určí body Nyquistovy frekvenční charakteristiky mající nižší frekvencí než v předchozí metodě, proto následné určení modelu druhého řádu s dopravním zpožděním pokaždé proběhlo správně. Třetí použitá metoda, publikovaná v práci Automatic Tuning of PID Controllers based on Asymmetric Relay Feedback (Bernner, 2015), identifikuje soustavu modelem prvního řádu s dopravním zpožděním, a ten byl i zde vypočítán ve všech případech. Přesnost určených modelů je porovnána v závěru práce. Jako nejlepší se ukázala druhá metoda, která oproti třetí metodě určila ve všech případech model přesněji a nikdy u ní neselhal výpočet parametrů modelu.

Klíčová slova: reléová zpětnovazební identifikace, dopravní zpoždění, normalizované dopravní zpoždění

Vedoucí: prof. Ing. Milan Hofreiter, CSc.

Abstract

In this diploma thesis, scripts utilising three different methods of relay feedback identification were written for the Tecomat Foxtrot PLC. The methods were chosen for the simplicity of measurements and following calculations. The first one (in Hofreiter 2017: Biased-Relay Feedback Identification for Time Delay Systems) was able to find points of the Nyquist's frequency characteristic. However, when estimating the parameters of second order time delayed model, the time delay estimate was smaller than zero in some cases, and thus meaningless. The second method (in Hofreiter 2018: Alternative Identification Method using Biased Relay Feedback) is a modification of the former. The frequency points of the Nyquist's frequency characteristic were estimated to be lower than with the first method; therefore the second order time delayed model was correctly identified in all cases. The third method (in Bernner 2015: Automatic Tuning of PID Controllers based on Asymmetric Relay Feedback) identifies the system using a first order time delayed model, which has been also identified in all cases. The accuracy of the identified models is discussed in the Conclusion section: the second method performed the best, because it did not fail in model parameters calculation in any of the attempts, and has been more accurate than the third one.

Keywords: relay feedback identification, time delay, normalized time delay

Title translation: Relay identification for PLC Tecomat Foxtrot

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Otázka identifikace soustav bude aktuální, dokud bude aktuální problematika jejich řízení. Pro přesné řízení soustavy je většinou třeba znát její model. Jednou z mnoha cest k jeho určení je reléová identifikace v uzavřeném regulačním obvodu. Ta má oproti jiným identifikačním metodám tu výhodu, že po dobu identifikace je systém řízen a riziko nežádoucí reakce systému na akční veličinu, jako je například dosažení technologicky nepřípustného stavu, je tak minimalizováno. Reléová identifikace v uzavřeném regulačním obvodu je metodou identifikace vhodnou pro soustavy s rychlou odezvou, jako je například reakce výkonu čerpadla na změnu napájecího napětí, kde s rostoucí dobou odezvy soustavy narůstá doba identifikace.

Problematika reléové identifikace má dlouhou historii a je popsaná v mnoha pracích. Frekvenční chování soustav k nastavení regulátorů použili už Ziegler a Nichols (1943) [1]. První metodu reléové identifikace popsal Tsypkin (1974) [2]. A měl řadu následovníků. O velký rozvoj se postarali Äström a Hägglund [3], na ně navázala Berner [4]. Novou metodu reléové identifikace publikoval Hofreiter [5]. Řada metod je také popsaná v publikaci Relay Autotuning for Identification and Control [6].

U identifikace soustav je však třeba teorie aplikovat v praxi. Pro průmyslové nasazení je nezbytné, aby identifikace probíhala samostatně a byla spolehlivá. Dále je nutné zvolit univerzální hardwarové řešení. Takovým univerzálním řešením mohou být programovatelné automaty. Pro účely této práce byl vybrán programovatelný automat Tecomat Foxtrot.

Pro svou jednoduchost byly k naprogramování pro programovatelný automat vybrány metody releové identifikace popsané v Biased-Relay Feedback Identification for Time Delay Systems [7], Alternative Identification Method using Biased Relay Feedback [8] a Automatic Tuning of PID Controllers based on Asymmetric Relay Feedback [9]. Zvolené metody jsou vhodné k identifikaci sytémů s jedním vstupem a jedním výstupem (Single input single output – SISO). První metodou je určení parametrů modelu ze tří bodů Nyquistovy frekvenční charakteristiky [7]. Z průběhů akční veličiny a regulované veličiny se určí tři body Nyquistovy frekvenční charakteristiky. Z nich se následně vypočítají parametry modelu druhého řádu s dopravním zpožděním. Druhou metodou je určení parametrů modelu ze tří bodů Nyquistovy frekvenční charakteristiky s přidaným dopravním zpožděním [8]. Její princip je téměř stejný jako u předchozí metody. Výstupem je také model druhého řádu s dopravním zpožděním, jen přepnutí relé je vždy opožděno. Třetí metodou je určení modelu pro

nastavení parametrů PI, PD nebo PID regulátorů [9]. Z parametrů relé a časů, kdy je relé v horní poloze a kdy je v spodní poloze, se vypočítají parametry modelu prvního řádu s dopravním zpožděním.

Část I

Programovatelný automat Tecomat Foxtrot CP-1015

Kapitola 2

Možnosti programovatelného automatu Tecomat Foxtot CP-1015

Firma Teco a.s. se sídlem v Kolíně je dlouholetým českým výrobcem programovatelných automatů, tedy zařízení spadajících do kategorie PLC (Programmable Logic Controller). Jejich vývoj, výroba i testování podléhají mezinárodnímu standardu IEC EN 61 131-3. Mezi jejich produkty patří modulární řídicí systém TECO TC700 pro střední a velké průmyslové aplikace, řada malých modulárních řídicích systémů Tecomat Foxtrot určených k montáži na DIN lištu do průmyslových rozvaděčů nebo také široké spektrum periferních zařízení s komunikací po kabelech i s bezdrátovou radiovou komunikací. Firma Teco také vyvíjí datové sběrnice pro komunikaci mezi základními moduly, dalšími moduly a periferními zařízeními.

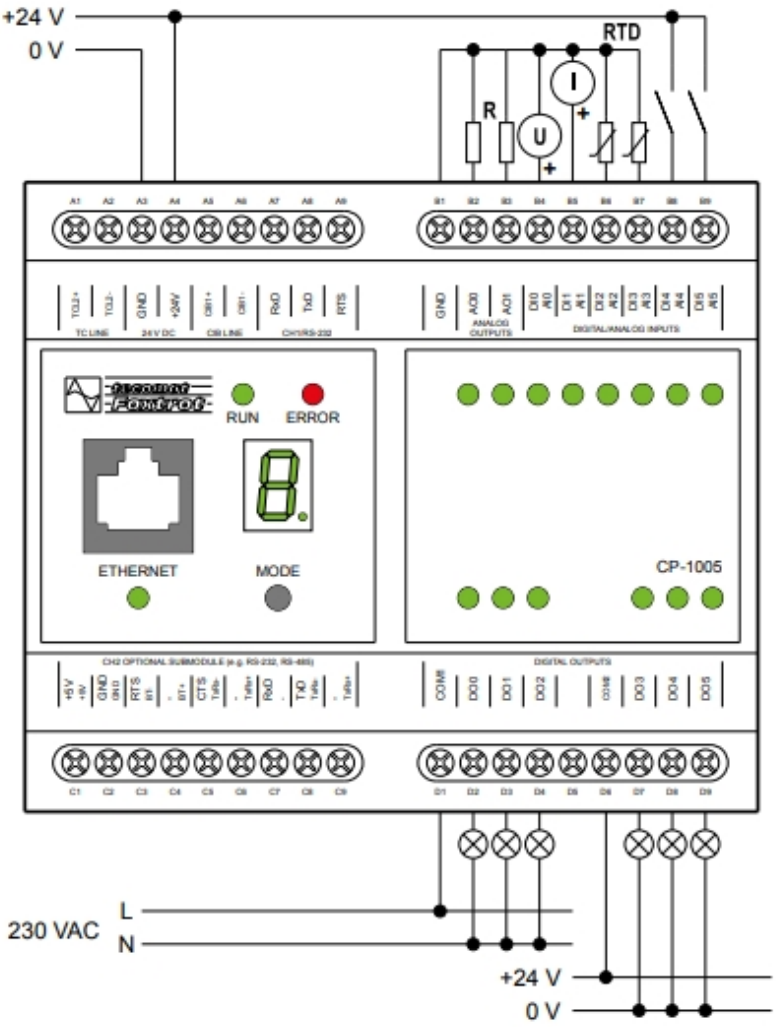
2.1 Hardware

Základní modul Tecomat Foxtrot CP-1015 je jeden z menších modulů v nabídce firmy Teco a.s.. Je vybaven šesti tlačítky a LCD displejem se čtyřmi řádky o dvaceti znacích. To je jeho jediná odlišnost od modulů řady CP-1005. Komunikovat může pomocí Ethernetu 100, CIB, TCL2 a RS-232.

2.1.1 Centrální jednotka

Centrální programovací jednotka je tvořena 32bitovým RISC procesorem. Doba cyklu PLC je 0.2 $ms/1$ k instrukcí. Je vybavena hodinami reálného času RTC. Paměť udrží prodram po dobu 500 h a v případě použití baterie 20 000 h. Paměť programu má velikost 192 KB, paměť proměnných 64 KB a 512 KB je velká interní paměť proměnných. Modul je vybaven SD/MMC slotem. Na SD kartu je možné ukládat data z měření a také se na ní ukládá webové rozhranní pro ovládání programu PLC.

2.1.2 Vstupy a výstupy



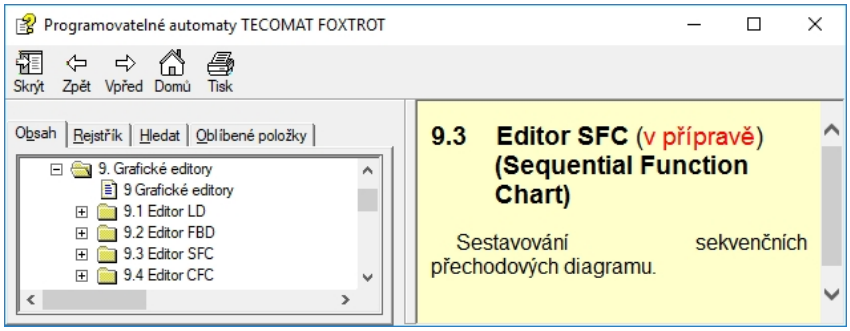
Obrázek 2.1: Schéma zapojení PLC Tecomat Foxtrot CP-1015 a CP-1015 (Převzato z technické dokumentace [10]).

Modul je vybaven šesti reléovými a dvěma analogovými výstupy. Šest analogových vstupů lze zapnout jako vstupy digitální. Při připojení dalších modulů je možné počet vstupů a výstupů výrazně navýšit. Například při použití komunikace po sériové sběrnici TCP2 až na celkový počet 134. Pro naši aplikci nebylo třeba připojovat žádné další moduly. Byl využíván pouze jeden analogový vstup a jeden analogový výstup.

2.2 Software

Pro programování PLC Tecomat Foxtrot a dalších PLC firmy Teco a.s. Kolín je volně dostupný software Mosaic. Program k dispozici od roku 2000 ve čtyřech jazykových verzích – české, anglické, ruské a německé. Pro odemčení plné verze programu je však třeba k počítači, na kterém se používá, připojit hardwarový klíč nebo do PLC nahrát softwarový klíč, který je pro každé PLC unikátní. Pro potřeby této a dalších prací byl firmou Teco a.s. Kolín poskytnut Ústavu přístrojové a řídicí techniky Fakulty strojní ČVUT v Praze zdarma softwarový klíč pro používané PLC Tecomat Foxtrot CP-1015. Programy popsané v této práci byly napsány ve verzi programu 2017.2 z února 2017.

V programu Mosaic je možné pracovat podle standardu IEC EN 61 131-3. Nabízí výběr mezi pěti programovacími jazyky, dvěma textovými a třemi grafickými. Vzhledem k tomu, že v nápovědě je připravený oddíl i pro další grafický jazyk (Sequential Function Chart – SFC), lze očekávat, že se v budoucnosti nabídka podporovaných jazyků rozšíří (obr. 2.2). Jazyky mají anglické názvy, jejich české ekvivalenty nedosahují platnosti odborných termínů. Při psaní programů je možné jazyky kombinovat. V prostředí Mosaic jsou k dispozici knihovny funkčních bloků.



Obrázek 2.2: Nápověda pro jazyk SFC (Převzato z nápovědy programu Mosaic [11]).

2.2.1 Ladder diagram

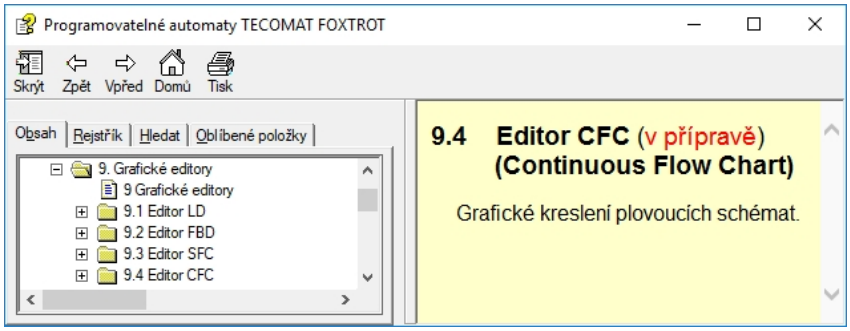
Ladder diagram (LD), neboli jazyk příčkového diagramu, je grafický programovací jazyk. Pro zapsání programů používá nákresy reléové logiky. Mezi dvěma napájecími sběrnicemi jsou příčky, ty se mohou větvit a jsou na nich zapojeny spínače, cívky, funkce a funkční bloky. Jednotlivé větve příček mohou být sepnuté nebo rozepnuté podle polohy spínačů.

2.2.2 Function block diagram

Function block diagram (FBD), jinak jazyk funkčního blokového schématu, vyjadřuje chování funkcí, funkčních bloků a programů pomocí propojených grafických bloků.

2.2.3 Continuous flow chart

Grafické kreslení plovoucích schémat (CFC) je poměrně novou variantou programování PLC a zatím nemá ani příslušné informace v nápovědě programu Mosaic (obr. 2.3). Svou strukturou je blízké jazyku funkčních bloků. Jako jediný z podporovaných jazyků neodpovídá standardu IEC EN 61 131-3.



Obrázek 2.3: Nápověda pro jazyk SFC (Převzato z nápovědy programu Mosaic [11]).

2.2.4 Instruction list

Instruction list (IL), česky jazyk seznamu instrukcí, je textový programovací jazyk. Svou strukturou a příkazy připomíná programovací jazyk Assembler.

2.2.5 Structured text

Structured text (ST), česky jazyk strukturovaného textu, je vyšší programovací jazyk vycházející z jazyků Ada, Pascal a C. Umožňuje používat iterační smyčky, jako je FOR a WHILE, nebo větvení podmínkami IF a CASE OF. Je vhodný pro programování nových funkčních bloků. V této práci byl použit právě jazyk Structured text z důvodu možnosti vytváření cyklů a kontrolních podmínek a také pro jednodušší provádění složitějších výpočtů [12].

Část II

Použité metody reléové identifikace

Kapitola 3

Určení parametrů modelu ze tří bodů Nyquistovy frekvenční charakteristiky

3.1 Základní princip metody

Tato metoda reléové identifikace je postavena na správném určení tří bodů nyquistovy frekvenční charakteristiky, z nichž jeden odpovídá frekvenci $\omega_0 = 0\ rad/s^{-1}$ a udává tedy statickou citlivost soustavy – K. Jeden z principů měření frekvenčních charakteristik spočívá v připojení harmonicky kmitajícího signálu o frekvenci ω na vstup do soustavy – u. Průběh této akční veličiny se zaznamenává stejně jako průběh výstupu ze soustavy – y. Ze zaznamenaných průběhů se pak určí jeden bod frekvenčních charakteristik soustavy. V Nyquistově frekvenční charakteristice je bod určen třemi parametry – frekvencí ω, reálnou souřadnicí $Re(G(j\omega))$ a imaginární souřadnicí $Im(G(j\omega))$. Případně může být bod dán trojicí parametrů – frekvencí ω, amplitudovým poměrem A a fázovým posuvem ϕ. Přepočet mezi parametry můžeme provést pomocí goniometrických funkcí. Uvažujeme-li, že máme bod určen pomocí reálné a imaginární souřednice, pak můžeme z těchto dvou parametrů vypočítat model soustavy o dvou neznámých parametrech. Pokud známe statickou citlivost soustavy – K, pak můžeme určit například model soustavy druhého řádu nebo prvního řádu s dopravním zpožděním. S každým dalším měřením se určí další bod Nyguistovy frekvenční charakteristiky, a tak vzniká možnost použít složitější model a nebo upřesnit jednodušší model.

Metodu, jak získat z jednoho měření dva body frekvenční charakteristiky, publikoval ve své práci Shifting Method for Relay Feedback Identification (2016) Milan Hofreiter [5]. Při znalosti statické citlivosti a dalších dvou bodů frekvenční charakteristiky by mělo být možné určit model soustavy daný statickou citlivostí a dalšími čtyřmi parametry. Takovým modelem s pěti parametry je anisochronní model

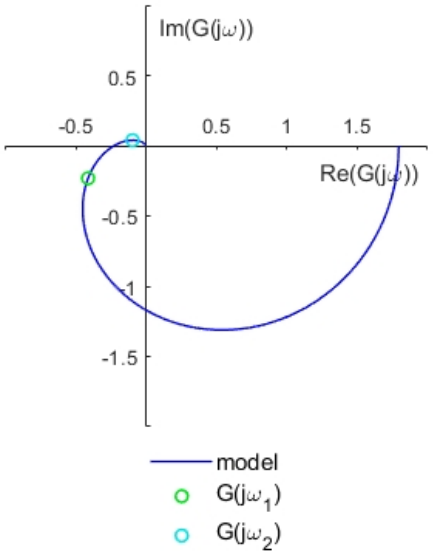
$$G_a(s) = \frac{Ke^{-T_d s}}{(\tau_1 s + 1)(\tau_2 s + e^{-\tau_y s})}, \tag{3.1}$$

kde K je statická citlivost, T_d je dopravní zpoždění, τ_1 a τ_2 jsou časové konstanty a τ_y je zpoždění hlavní zpětné vazby. Tímto modelem je možné popsat s vysokou přesností i složité soustavy. Avšak vzhledem k nelineárnosti modelu je potřeba k výpočtu jeho parametrů použít některou z numerických metod výpočtu. Tomuto tématu jsem se blíže věnovala v bakalářské práci Odhad parametrů anisochronního

modelu ze zadaných bodů frekvenční charakteristiky [13]. Lepších výsledků pak bylo dosaženo v práci Určení parametrů anisochronního modelu soustavy pomocí releové identifikace a diferenciální evoluce [14]. Vzhledem k náročnosti výpočtů a nutnosti použít jako výpočetní jednotku PLC s omezenou pamětí byl vybrán jednodušší model a to model druhého řádu s dopravním zpožděním (Second order time delayed model – SOTD)

$$G(s) = \frac{Ke^{-T_d s}}{a_2 s^2 + a_1 s + 1}, \tag{3.2}$$

kde K je statická citlivost, T_d je dopravní zpoždění a a_1 a a_2 jsou časové konstanty. Pro určení parametrů tohoto jednoduššího modelu není třeba použít numerické metody výpočtu. Matematické vztahy k přímému výpočtu parametrů modelu ze dvou bodů Nyquistovy frekvenční charakteristiky a statické citlivosti byly publikovány v práci Biased-Relay Feedback Identification for Time Delay Systems [7].

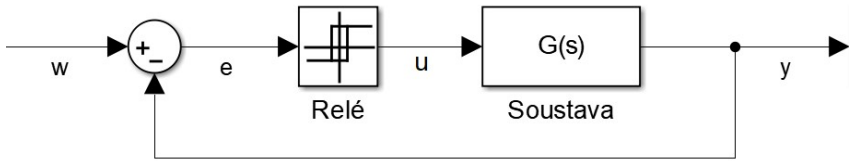


Obrázek 3.1: Nejčastější rozložení bodů Nyquistovy frekvenční charakteristiky určených identifikací.

Nevýhodou metody je, že nalezené body frekvenční charakteristiky leží ve druhém a třetím kvadrantu (obr. 3.1). O průběhu Nyquistovy frekvenční charakteristiky soustavy ve čtvrtém kvadrantu tak nemáme zádné informace. Pro určení modelu je tedy vhodnější nalézt body frekvenční charakteristiky s fází menší než π *rad*.

3.2 Zapojení soustavy pro identifikaci

Pro určení dvou bodů Nyquistovy frekvenční charakteristiky se použije zpětnovazební zapojení s relé (obr.3.2),



Obrázek 3.2: Schéma základního zapojení pro reléovou identifikaci.

kde w je žádaná hodnota, e je regulační odchylka, y je regulovaná veličina a u je akční veličina.

3.3 Přepínání relé

K identifikaci soustavy touto metodou se používá asymetrické relé s horní mezí hystereze h_1 a dolní mezí hystereze h_2. Při známém pracovním bodě daném hodnotou akční veličiny u_0 a hodnotou regulované veličiny y_0 připojíme relé do uzavřeného regulačního obvodu soustavy. Přepínání relé mezi horní u_1 a dolní polohou u_2 probíhá podle pravidel

$$u(t) = \begin{cases} u_1 \quad pokud \quad y(t) < y_0 - h_2 \\ u_1 \quad pokud \quad y(t) < y_0 + h_1,\ u(t^{-1}) = u_1 \\ u_2 \quad pokud \quad y(t) > y_0 + h_1 \\ u_2 \quad pokud \quad y(t) > y_0 - h_2,\ u(t^{-1}) = u_2 \end{cases}, \tag{3.3}$$

kde $u(t^{-1})$ je hodnota akční veličiny v předcházejícím okamžiku, h_1 je horní mez hystereze relé a h_2 je dolní mez hystereze relé. Perioda kmitu se pak vypočte

$$t_p = t_1 + t_2, \tag{3.4}$$

kde t_1 je čas, kdy je relé v horní poloze, a t_2 je čas, kdy je relé v dolní poloze. Identifikace nemůže proběhnout správně, pokud neplatí

$$t_1 < t_2. \tag{3.5}$$

3.4 Určení bodů Nyquistovy frekvenční charakteristiky

3.4.1 První bod Nyquistovy frekvenční charakteristiky

První určovaný bod Nyquistovy frekvenční charakteristiky je bod s úhlovou frekvencí $\omega_0 = 0\ rad \cdot s^{-1}$. Z teorie frekvenčního chování soustav je zřejmé, že tento bod má nulovou imaginární souřadnici a velikost jeho reálné souřadnice udává statickou citlivost soustavy – K. Pro statickou citlivost při použití asymetrického relé a ustálených kmitech akční veličiny platí

$$K = \frac{I_y}{I_u}, \tag{3.6}$$

kde

$$I_y = \int_{t_p} (y(t) - y_0)dt \tag{3.7}$$

$$I_u = \int_{t_p} (u(t) - u_0)dt = u_1 t_1 + u_2 t_2. \tag{3.8}$$

Dosazením do (3.6) z (3.7) a (3.8) se vypočítá hodnota statické citlivosti soustavy. Je však třeba oba průběhy přepočítat tak, aby nula byla vždy v hodnotě odpovídající pracovnímu bodu [5].

Druhou možnou cestou k určení statické citlivosti soustavy je výpočet ze statické charakteristiky. Stačí znát její dva body $[u_{K1}, y_{K1}]$ a $[u_{K2}, y_{K2}]$ blízké pracovnímu bodu. Pak dosazením do vztahu

$$K = \frac{y_{K2} - y_{K1}}{u_{K2} - u_{K1}} \tag{3.9}$$

vypočítáme statickou citlivost soustavy.

3.4.2 Druhý bod Nyquistovy frekvenční charakteristiky

Pro frekvenci druhého bodu Nyquistovy frekvenční charakteristiky platí

$$\omega_u = \frac{2\pi}{t_p}. \tag{3.10}$$

Bod charakteristiky pak určíme ze vztahu

$$G(j\omega_u) = \frac{y_A}{A_u} e^{j\phi_{uy}}, \tag{3.11}$$

kde

$$A_u = \frac{2H}{\pi} sin(\phi_u), \tag{3.12}$$

$$H = u_1 + |u_2|, \tag{3.13}$$

$$\phi_u = \frac{\omega_u \cdot t_1}{2} = \frac{\pi \cdot t_1}{t_p}, \tag{3.14}$$

a fázový posun mezi akční a regulovanou veličinou pak je

$$\phi_{uy} = -\omega_u \cdot t_{uy}. \tag{3.15}$$

Bod Nyquistovy frekvenční charakteristiky je možné určit i bez znalosti amplitud kmitů nebo fázového posunu. Stačí změřit periodu kmitů a z ní vypočítat jejich frekvenci (3.10). Pak je možné zjistit bod frekvenční charakteristiky pomocí podílu integrálů ze vstupu a výstupu přes jednu periodu

$$G(j\omega_u) = \frac{\int_t^{t+t_p} y(\tau) e^{-j\omega_u \tau} d\tau}{\int_t^{t+t_p} u(\tau) e^{-j\omega_u \tau} d\tau}, \tag{3.16}$$

kde t je čas počátku ustálené periody [5].

3.4.3 Třetí bod Nyquistovy frekvenční charakteristiky

Základním předpokladem pro určení třetího bodu Nyquistovy frekvenční charakteristiky je možnost popsat soustavu lineárním modelem. Výpočet třetího bodu frekvenční charakteristiky je pak velmi podobný výpočtu druhého bodu. Je však nutné vypočítat novou frekvenci ω_{u2} a této frekvenci odpovídající hodnoty vstupu u_a a výstupu y_a soustavy. Frekvence třetího bodu je dvojnásobná než frekvence druhého bodu

$$\omega_{u2} = 2 \cdot \omega_u. \tag{3.17}$$

Průběhy akční a regulované veličiny jsou pak tvořeny součtem původního průběhu se stejným průběhem posunutým o půl periody [5]. Platí pro ně vztahy

$$u_a(t) = u(t) + u\left(t - \frac{t_p}{2}\right) \tag{3.18}$$

a

$$y_a(t) = y(t) + y\left(t - \frac{t_p}{2}\right). \tag{3.19}$$

Bod frekvenční charakteristiky je dán vztahem

$$G(j\omega_{u2}) = \frac{\int_t^{t+t_p} y_a(\tau) e^{-j\omega_{u2}\tau} d\tau}{\int_t^{t+t_p} u_a(\tau) e^{-j\omega_{u2}\tau} d\tau}. \tag{3.20}$$

Integrace zde tedy probíhá přes dvě periody nově spočtených signálů. Integrace přes více period může zpřesnit polohu nalezených bodů frekvenčních charakteristik.

3.5 Parametry modelu

Výpočet parametrů modelu druhého řádu s dopravním zpožděním (3.2) je dán vztahy popsanými v [5]. Z bodů Nyquistovy frekvenční charakteristiky daných frekvencí a souřadnicemi v komplexním prostoru se vypočítají časové konstanty a_1, a_2 a velikost dopravního zpoždění T_d.

$$a_2 = \frac{1}{2 \cdot \omega_u^2} \sqrt{\frac{1}{3}\left(\frac{K^2}{|G(j\omega_{u2})|} - \frac{4 \cdot K^2}{|G(j\omega_u)|} + 3\right)} \tag{3.21}$$

$$a_1 = \frac{1}{\omega_u} \sqrt{\frac{K^2}{|G(j\omega_u)|} - (1 - a_2 \cdot \omega_u^2)^2} \tag{3.22}$$

$$T_d = \frac{1}{2}\left(T_u + T_{u2}\right) \tag{3.23}$$

$$T_u = \frac{\frac{1}{a_2 \cdot (j\omega_u)^2 + a_1 \cdot j\omega_u} - arg(G(j\omega_u))}{\omega_u} \tag{3.24}$$

$$T_{u2} = \frac{\frac{1}{a_2 \cdot (j\omega_{u2})^2 + a_1 \cdot j\omega_{u2}} - arg(G(j\omega_{u2}))}{\omega_{u2}} \tag{3.25}$$

Těmito parametry je určen SOTD model soustavy.

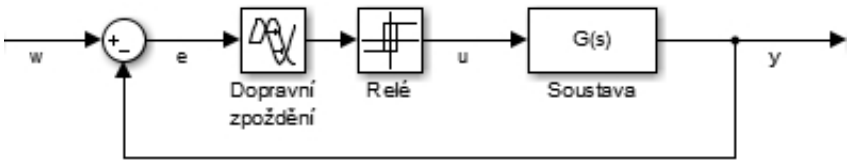
Kapitola 4

Určení parametrů modelu ze tří bodů Nyquistovy frekvenční charakteristiky s přidaným dopravním zpožděním

Tento způsob identifikace je novější vylepšenou verzí reléové identifikace popsané v předchozí kapitole.

4.1 Měření

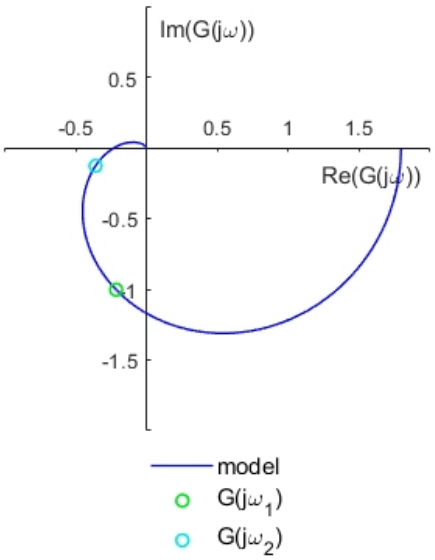
Měření probíhá na podobném principu jako u metody v předchozí kapitole. Je však třeba provést drobnou úpravu zapojení. Do předchozího zapojení se vloží dopravní zpoždění tak, aby relé na dosažení meze hystereze reagovalo se zpožděním. Ve schématu zapojení (obr. 4.1) pak w udává žádanou veličinu (požadovaný výstup v pracovním bodě), e je regulační odchylka (odchylka regulované veličiny od pracovního bodu), u je akční veličina a y je regulovaná veličina.



Obrázek 4.1: Schéma zapojení pro reléovou identifikaci s přidaným dopravním zpožděním.

4.2 Výpočet

Pro výpočet statické citivosti K soustavy a polohy dvou bodů Nyquistovy frekvenční charakteristiky $G(j\omega_1)$ a $G(j\omega_2)$ se použijí vztahy (3.7), (3.8), (3.6), (3.10), (3.16), (3.17), (3.20) a stejně jako u předchozí verze releové identifikace z bodů Nyquistovy frekvenční charakteristiky. V závislosti na velikosti zvoleného dopravního zpoždění se body frekvenční charakteristiky posunou ze druhého a třetího kvadrantu směrem ke kvadrantu čtvrtému (obr. 4.2).



Obrázek 4.2: Nejčastější rozložení identifikací určených bodů Nyquistovy frekvenční charakteristiky.

Pro výpočet parametrů modelu byla zvolena cesta popsaná v práci Biased-Relay Feedback Identification for Time Delay Systems [7], stejně jako u jednodušší metody reléové identifikace pomocí tří bodů Nyquistovy frekvenční charakteristiky. Použijí se tedy vztahy (3.21), (3.22), (3.24), (3.25) a (3.23).

Kapitola 5

Určení modelu pro nastavení parametrů PI, PD a PID regulátorů

5.1 Základní princip metody

Pro správné nastavení parametrů regulátoru, a tedy pro dobrou regulaci, je potřeba znát dostatečně přesný model řízené soustavy. K určení modelu soustavy je nutné použít některou z metod identifikace. V práci Automatic Tuning of PID Controllers based on Asymmetric Relay Feedback [9] je popsaná metoda reléové identifikace v uzavřeném regulačním obvodu a následný výpočet parametrů PID, PI a PD regulátorů z určeného modelu. Identifikace je zde tedy mezikrokem při nastavování regulátorů. Soustavu je možné identifikovat modelem prvního řádu s dopravním zpožděním (First order time delayed model – FOTD), druhého řádu s dopravním zpožděním (Second order time delayed model – SOTD) a integračním modelem s dopravním zpožděním (Integrating time delayed model – ITD) a modelem prvního řádu s integrací s dopravním zpožděním (Integrating plus first order time delayed model – IFOTD). Pro výpočet parametrů většiny uvedených modelů by bylo potřeba použít numerické metody výpočtu. Vzhledem k omezené paměti PLC je více než vhodné se jejich použití vyvarovat. Tím se zúžuje výběr na modely FOTD a ITD. Pro výpočet parametrů těchto modelů byly publikovány matematické vztahy [9]. Vzhledem k nižší pravděpodobnosti výskytu integračního systému byl pro identifikaci vybrán model FOTD.

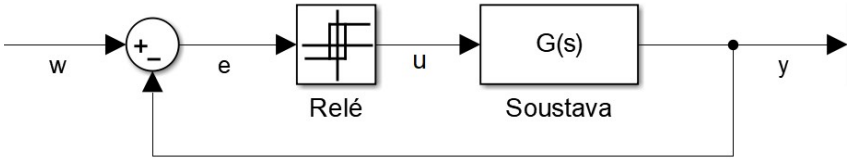
Soustava bude tedy identifikována modelem prvního řádu s dopravním zpožděním

$$G(s) = \frac{Ke^{-T_d s}}{Ts + 1}, \tag{5.1}$$

kde K je statická citlivost, T_d je dopravní zpoždění a T je časová konstanta. Z nastavených parametrů relé a časů, kdy relé setrvává v horní poloze a kdy zůstává ve spodní poloze, se vypočítá normalizované dopravní zpoždění τ a statická citlivost soustavy – K. Z nich se následně vypočte velikost časové konstanty a dopravního zpoždění.

5.2 Reléová identifikace

Reléová identifikace dle Josefin Berner [9] probíhá v uzavřeném regulačním obvodu v zapojení podle schématu na obr. 5.1.



Obrázek 5.1: Schéma základního zapojení pro reléovou identifikaci.

Zde je w žádaná hodnota, e je regulační odchylka, y je regulovaná veličina a u je akční veličina.

5.3 Přepínání relé

Na počátku identifikace se systém nachází v pracovním bodě daném hodnotou akční veličiny u_0 a regulované veličiny y_0. Asymetrické relé přepíná mezi horní polohou u_{on} a dolní polohou u_{off} podle předpisu

$$u(t) = \begin{cases} u_{on} & pokud \quad y(t) < y_0 - h \\ u_{on} & pokud \quad y(t) < y_0 + h,\ u(t^{-1}) = u_{on} \\ u_{off} & pokud \quad y(t) > y_0 + h \\ u_{off} & pokud \quad y(t) > y_0 - h,\ u(t^{-1}) = u_{off} \end{cases}, \tag{5.2}$$

kde $u(t^{-1})$ je hodnota akční veličiny v předcházejícím okamžiku a h je hystereze relé. Aby se zajistila asymetričnost relé, je vhodné dodržet podmínku pro stupeň asymetrie relé

$$\gamma = \frac{max(d_1, d_2)}{min(d_1, d_2)} > 0, \tag{5.3}$$

přičemž

$$d_1 = u_{on} - u_0 \tag{5.4}$$

a

$$d_2 = u_0 - u_{off}. \tag{5.5}$$

Nedodržením podmínky pro asymetrii relé by došlo k chybě při výpočtu statické citlivosti soustavy – K.

5.4 Normalizované dopravní zpoždění

Normalizované dopravní zpoždění τ je důležitým parametrem při nastavování PID regulátorů [15]. Jednou z možností, jak ho určit, je právě použití reléové identifikace v

uzavřeném regulačním obvodu. Normalizované dopravní zpoždění je možné vypočítat ze stupně asymetrie relé (5.3) a indexu poloviny periody

$$\rho = \frac{max(t_{on}, t_{off})}{min(t_{on}, t_{off})}, \tag{5.6}$$

kde t_{on} je doba, po kterou je relé v horní poloze, a t_{off} je doba, po kterou je relé v dolní poloze. Pro normalizované dopravní zpoždění platí

$$\tau = \frac{\gamma - \rho}{(\rho - 1)(0,35\rho + 0,65)}. \tag{5.7}$$

V případě, že je normalizované dopravní zpoždění τ malé, jsou časové úseky t_{on} a t_{off} podobné délky, i přestože jsou amplitudy rozdílné. Pro normalizované dopravní zpoždění musí platit

$$0 \leq \tau \leq 1. \tag{5.8}$$

Krajní hodnoty normalizovaného zpoždění vycházejí v případech uvedených v tebulce 5.1.

ρ	γ	τ
1	γ	1
γ	γ	0

Tabulka 5.1: Vliv hodnot ρ a γ na τ [15].

5.5 Určení modelu

Při určování parametrů FOTD modelu (5.1) se ze zaznamenaných průběhů akční a regulované veličiny vypočítá statická citlivost K, časová konstanta T a velikost dopravního zpoždění T_d.

Statická citlivost soustavy se vypočte z integrálů z akční veličiny (5.10) a regulované veličiny (5.11) na jedné periodě kmitání relé t_p.

$$t_p = t_{on} + t_{off} \tag{5.9}$$

$$I_u = \int_{t_p} (u(t) - u_0)dt = u_{on}t_{on} + u_{off}t_{off} \tag{5.10}$$

$$I_y = \int_{t_p} (y(t) - y_0)dt. \tag{5.11}$$

Pro statickou citlivost platí

$$K = \frac{I_y}{I_u}. \tag{5.12}$$

Dosazením ze vztahů (5.10) a (5.11) do (5.12) dostaneme hodnotu statické citlivosti modelu. Jedná se o stejný způsob určení statické citlivosti, jaký je použitý u metody reléové identifikace ze tří bodu Nyquistovy frekvenční charakteristiky [5], [8].

Použití symetrického rélé by mělo za následek, že by ze vztahu (5.10) vyšlo $I_u = 0$. Pak by ve vztahu pro výpočet statické citlivosti (5.12) bylo třeba dělit nulou. Pokud

tedy chceme určovat statickou citlivost ze vztahu (5.12), musíme dodržet podmínku asymetričnosti relé. I u asymetrického relé může nastat stav kdy $I_u = 0$. Stane se tak však pouze v případě, že $u_{on}t_{on} = -u_{off}t_{off}$. To by naznačovalo, že se jedná o integrační proces, a v takovém případě by bylo lepší použít raději ITD model

$$G(s) = \frac{Ke^{-T_d s}}{s} \tag{5.13}$$

než FOTD model. K výpočtu velikosti dopravního zpoždění T_d a časové konstanty T FOTD modelu můžeme použít následující vztahy [9]:

$$t_{on} = T\,ln\left(\frac{\frac{h}{|K|} - d_2 + e^{\frac{T_d}{T}}(d_1 + d_2)}{d_1 - \frac{h}{|K|}}\right) \tag{5.14}$$

$$t_{off} = T\,ln\left(\frac{\frac{h}{|K|} - d_1 + e^{\frac{T_d}{T}}(d_1 + d_2)}{d_2 - \frac{h}{|K|}}\right). \tag{5.15}$$

Takovou soustavu rovnic ale není možné vypočítat analytickými metodami. Pokud ale připojíme k výpočtu vztah pro normalizované dopravní zpoždění soustav prvního řádu s dopravním zpožděním

$$\tau = \frac{T_d}{T_d + T}, \tag{5.16}$$

pak nebudeme muset použít numerické metody výpočtu. Pokud z (5.7) známe velikost normalizovaného dopravního zpoždění, pak můžeme upravit vztah (5.16) na

$$\frac{T_d}{T} = \frac{\tau}{1 - \tau} \tag{5.17}$$

a vyčíslit jeho hodnotu. Vyjádřením časové konstanty T z (5.14) dostaneme vztah

$$T = \frac{t_{on}}{ln\left(\frac{\frac{h}{|K|} - d_2 + e^{\frac{T_d}{T}}(d_1 + d_2)}{d_1 - \frac{h}{|K|}}\right)}. \tag{5.18}$$

Dosazením z (5.17) a (5.12) do (5.18) dostaneme hodnotu časové konstanty soustavy. Dopravní zpoždění vypočítáme z

$$T_d = T\frac{\tau}{1 - \tau}, \tag{5.19}$$

když do vztahu dosadíme výsledek z (5.18). Tím jsou určeny parametry FOTD modelu.

Část III

Programové řešení reléové identifikace

Kapitola 6

Určení parametrů modelu ze tří bodů Nyquistovy frekvenční charakteristiky

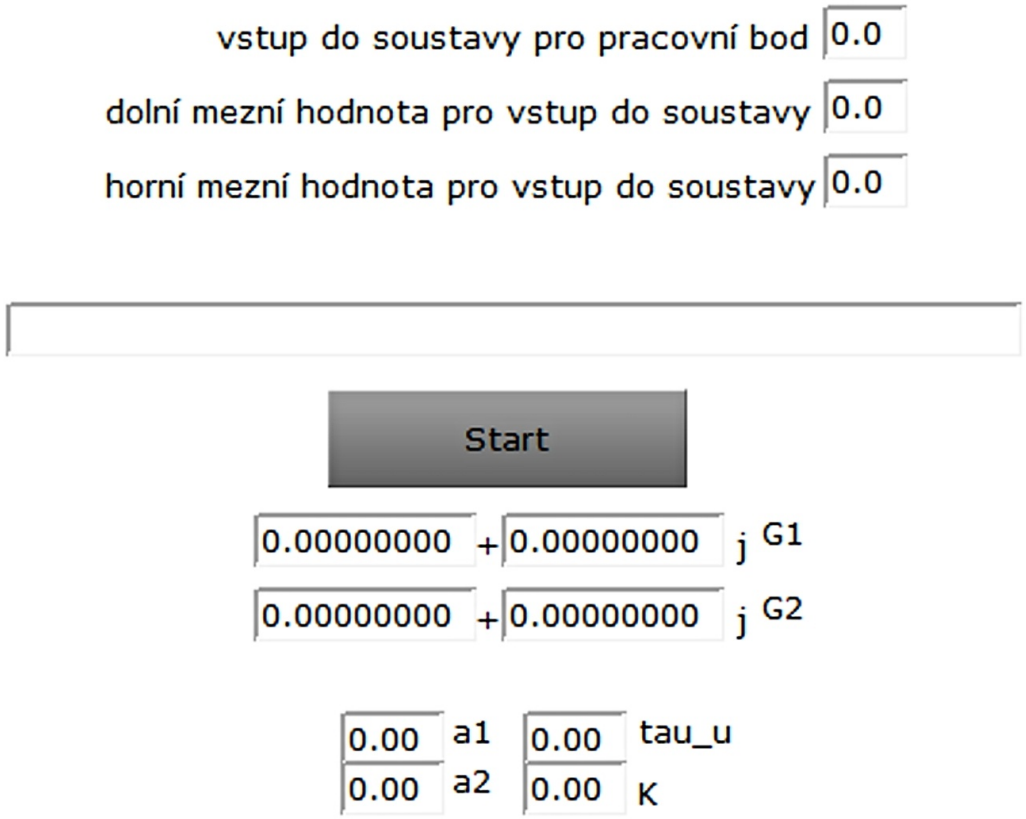
Vzhledem k principu zpracování programu v PLC byl program pro reléovou identifikaci napsán pomocí struktury CASE OF. Pravidelně se střídají fáze inicializační, ve kterých se nastavují hodnoty výstupu a pomocných proměnných do hodnot potřebných pro správnou funkci následující fáze, a fáze hlavní, ve které probíhá měření a výpočty. Tyto dvě fáze spolu tedy úzce souvisejí a je nutné provést obě.

6.1 Spuštění identifikačního procesu

6.1.1 CASE 00 a CASE 01 – kontrola zadaných údajů a spuštění identifikace

První část programu slouží k nastavení parametrů identifikace a jejímu spuštění. Jelikož se nacházíme na samotném začátku, není třeba v inicializační fázi (CASE 00) zadávat žádné počáteční hodnoty výstupu a vnitřních proměnných. Tato fáze je v programu pouze pro zachování struktury skriptu a přímo odkazuje na hlavní fázi (CASE 01).

V hlavní fázi pak vložený funkční blok $fbWebInputControl$, napsaný pro účely této práce, vyhodnocuje parametry regulace zadané uživatelem prostřednictvím webového rozhraní (obr. 6.1).



Obrázek 6.1: Webové rozhraní pro ovládání programu reléové identifikace.

Pro správný a bezpečný chod identifikace je třeba znát předpokládaný vstup do soustavy v pracovním bodě (u_0). V tomto pracovním bodě bude soustava identifikována a určený model bude reálné soustavě nejvíce odpovídat. Dále je třeba zadat horní mezní ($u_{MaxValue}$) a dolní mezní hodnotu ($u_{MinValue}$) vstupu do soustavy. Toto omezení je důležité, aby se soustava nedostala do stavu, který je nežádoucí, ať už z důvodu konstrukce soustavy nebo technologie, ke které soustava slouží. Pro určení těchto mezních hodnot vstupu i hodnoty vstupu v pracovním bodě je potřeba soustavu dobře znát a rozumět jejímu principu.

Po zadání těchto parametrů a stisknutí tlačítka *Start* dojde ke kontrole zadaných údajů. Při jejich správném zadání se přejde do další části programu a tlačítko zůstane neaktivní. Pokud jsou údaje zadány chybně, vypíše se uživateli chybová hláška (tab. 6.1) a tlačítko je aktivní. Pak musí uživatel zadat údaje korektně a znovu stisknout tlačítko *Start*.

chyba	chybová hláška
$u_{MinValue} > u_{MaxValue}$	horní mez musí ležet nad dolní mezí
$u_{MinValue} < 0$	horní a dolní mez musí ležet mezi 0 a 10
$u_{MaxValue} > 10$	horní a dolní mez musí ležet mezi 0 a 10
$(u_0 - 0,5) < u_{MinValue}$	vstup musí ležet mezi (dolní mezí + 0,5) a horní mezí
$u_0 > u_{MaxValue}$	vstup musí ležet mezi (dolní mezí + 0,5) a horní mezí

Tabulka 6.1: Chybové hlášky při špatném zadání parametrů regulace.

6.2 Určení pracovního bodu, statické citlivosti soustavy a velikosti šumu výstupního signálu

6.2.1 CASE 10 a CASE 11 – nastavení konstantní hodnoty vstupu a čekání na ustálený stav

V inicializační fázi se na vstup do soustavy nastaví hodnota $u = u_0 - 0,5$. Což je velikost akční veličiny v pracovním bodě snížená o $0,5$.

V hlavní fázi je pak volán funkční blok *fbstabilization*, který kontroluje ustálení výstupu ze soustavy, tedy regulované veličiny. Jmenovaný funkční blok byl napsán ve spolupráci s Františkem Hylmarem v rámci předmětu Projekt III. Jeho princip je založen na průměrování hodnoty výstupu po čas 10 s a porovnání tří po sobě naměřených průměrů. Pokud jsou tyto časové průměry stejné nebo je z nich zřejmé, že průběh výstupní veličiny změnil svoji monotónnost, pak je soustava považována za ustálenou. Po ustálení soustavy je tato fáze ukončena.

6.2.2 CASE 20 a CASE 21 – určení bodu blízkého pracovnímu bodu

Inicializační fáze poslouží pro vynulování proměnných, které budou použity v hlavní fázi.

Soustava je v ustáleném stavu z předchozí sekce programu. Proto můžeme určit hodnotu ustáleného výstupu. Po dobu 1 s přičítáme do pomocné proměnné při každém cyklu PLC aktuální hodnotu výstupu y. Zároveň počítáme kolik cyklů proběhlo. Vydělením těchto proměnných pak získáváme průměrnou hodnotu ustáleného výstupu y_{K1} při vstupu $u_{K1} = u_0 - 0.5$.

6.2.3 CASE 30 a CASE 31 – nastavení hodnoty vstupu do pracovního bodu a čekání na ustálený stav

V inicializační fázi se nastaví hodnota akční veličiny na $u = u_0$. Tedy do pracovního bodu.

V hlavní fázi je umístěn funkční blok pro kontrolu ustáleného stavu soustavy. Ten je popsaný v oddíle 6.2.1. Po ustálení soustavy je fáze ukončena.

6.2.4 CASE 40 a CASE 41 – určení pracovního bodu a statické citlivosti soustavy

Inicializační fáze poslouží pro vynulování proměnných, které budou použity v hlavní fázi.

Soustava je v ustáleném stavu z předchozí sekce programu. Proto můžeme určit hodnotu ustáleného výstupu. Po dobu 1 s přičítáme do pomocné proměnné při každém cyklu PLC aktuální hodnotu výstupu y. Zároveň počítáme, kolik cyklů proběhlo. Vydělením těchto proměnných pak získáváme průměrnou hodnotu ustáleného výstupu y_{K2} při vstupu $u_{K2} = u_0$. Průměrná hodnota výstupu soustavy při vstupu u_0 pak udává velikost výstupu v pracovním bodě $y_0 = y_{K2}$. Protože nyní již známe vstup a výstup soustavy ve dvou bodech (v pracovním bodě a bodě se vstupem menším o $0,5$), můžeme vypočítat statickou citlivost soustavy – K.

$$K = \frac{y_{K2} - y_{K1}}{u_{K2} - u_{K1}}. \tag{6.1}$$

6.2.5 CASE 50 a CASE 51 – určení maximální hodnoty šumu v pracovním bodě

Inicializační fáze poslouží pro vynulování proměnné, která se použije v hlavní fázi.

Po dobu 10 s se v hlavní fázi měří maximální absolutní výchylka výstupu soustavy od jeho průměrné hodnoty. Z takto určené maximální hodnoty šumu ($noise_{Max}$) se vypočtou meze hystereze relé.

$$hysteresis_{LowValue} = 10 \cdot noise_{Max}. \tag{6.2}$$

$$hysteresis_{HighValue} = 6 \cdot noise_{Max}. \tag{6.3}$$

6.3 Čekání na ustálení periody kmitu soustavy a vzorkování vstupu a výstupu

6.3.1 CASE 70 a CASE 71 – hledání ustálené periody kmitu

V inicializační fázi se vypočtou hodnoty vstupu do soustavy při horní a dolní poloze relé. Horní polohu relé $u_{HighValue}$ vypočteme jako

$$u_{HighValue} = u_0 + 5. \tag{6.4}$$

Pokud platí

$$u_{HighValue} > u_{MaxValue}, \tag{6.5}$$

pak

$$u_{HighValue} = u_{MaxValue}. \tag{6.6}$$

Dolní polohu relé $u_{LowValue}$ vypočteme jako

$$u_{LowValue} = u_0 - 3. \tag{6.7}$$

Pokud platí

$$u_{LowValue} < u_{MinValue}, \tag{6.8}$$

pak

$$u_{LowValue} = u_{MinValue}. \tag{6.9}$$

Následuje nastavení akční veličiny na hodnotu $u_{HighValue}$ a uložení aktuálního času do proměnných *leadingEdgeRtc* a *trailingEdgeRtc* udávajících čas, kdy došlo k náběžné a sestupné hraně. Vynulují se dočasné pomocné proměnné *tempBool* a *tempBool*2. Do čítače náběžných hran *leadingEdgeCounter* se uloží číslo 1 a do čítače sestupných hran *trailingEdgeCounter* číslo 0. Doby, kdy je relé v horní poloze T_1 a dolní poloze T_2, nastavíme na 1. Poté následuje hlavní fáze.

Pokud výstup soustavy překročí hodnotu $y_0 + hysteresis_{HighValue}$ a ještě není akční veličina natavena na $u_{LowValue}$, pak jí na tuto hodnotu nastavíme a aktuální čas si uložíme do proměnné *tempDataTime*. Hodnotu čítače sestupných hran *trailingEdgeCounter* zvýšíme o 1. Do proměnné *trailingEdgeRtc* uložíme aktuální čas, tedy čas sestupné hrany. Následně určíme hodnotu pomocné proměnné *tempReal* pomocí funkce SUB_DT_DT pro odčítání časů ve formátu datum a čas a rozdíl pak převedeme na reálné číslo pomocí funkce $TIME\_TO\_REAL$ z knihovny Mosaicu TimeLib.

$$tempReal = tempDateTime - leadingEdgeRtc \tag{6.10}$$

Následuje kontrola, zda poslední doba, kdy bylo relé v horní poloze, se od předchozí neodchyluje o více než 2 %. Když tuto podmínku splňuje, nastaví se pomocná proměnná *tempBool* na $TRUE$, v opačném případě jako $FALSE$. Pak do proměnné T_1 uložíme hodnotu pomocné proměnné *tempReal*.

Pokud výstup soustavy klesne pod hodnotu $y_0 - hysteresis_{LowValue}$ a ještě není vstup do soustavy nataven na $u_{HighValue}$, pak jej na tuto hodnotu nastavíme a aktuální čas si uložíme do proměnné *tempDataTime*. Hodnotu čítače náběžných hran *leadingEdgeCounter* zvýšíme o 1. Do proměnné *leadingEdgeRtc* uložíme aktuální čas, tedy čas náběžné hrany. Následně určíme hodnotu pomocné proměnné *tempReal* pomocí funkce SUB_DT_DT pro odčítání časů ve formátu datum a čas a rozdíl pak převedeme na reálné číslo pomocí funkce $TIME\_TO\_REAL$.

$$tempReal = tempDateTime - trailingEdgeRtc \tag{6.11}$$

Následuje kontrola, zda poslední doba, kdy bylo relé v dolní poloze, se od předchozí neodchyluje o více než 2 %. Když tuto podmínku splňuje, nastaví se pomocná proměnná *tempBool*2 jako $TRUE$, v opačném případě jako jako $FALSE$. Následně do proměnné T_2 uložíme hodnotu pomocné proměnné *tempReal*.

Nakonec provedeme kontrolu, zda jsou obě pomocné proměnné *tempBool* a *tempBool*2 nastaveny na hodnotu $TRUE$. Pokud tomu tak je, pak jsou periody kmitu ustálené a po vypočtení periody kmitu soustavy T_{period} se přejde do dalšího úseku identifikace.

$$T_{period} = T_1 + T_2 \tag{6.12}$$

6.3.2 CASE 80 a CASE 81 – vzorkování vstupu a výstupu soustavy

V inicializační fázi proběhne nejprve kontrola, zda je splněna podmínka

$$T_1 < T_2. \tag{6.13}$$

Pokud je splněná, pak se připraví pomocné proměnné pro další měření. Hodnota aktuálního indexu vzorkování $samplingArrayIndex$ a indexy posledních uložených vzorků v prvním sloupci $samplingArrayLastIndex[0]$, ve druhém sloupci $samplingArrayLastIndex[1]$ a ve třetím sloupci pole $samplingArrayLastIndex[2]$ se nastaví na vyšší hodnotu, než může reálně nastat. Konkrétně je to 1 000. Nastaví se také index ukazující do sloupce pole pro ukládání vzorků z příští periody kmitu $samplingArrayColumnIndex = 0$ a indexy vedlejšího sloupce pole $samplingArrayColumnNeighbour1Index = 1$ a druhého vedlejšího sloupce pole $samplingArrayColumnNeighbour2Index = 2$. Pomocná proměnná $tempCounter$ se vynuluje. Připraví se pomocné proměnné pro kontrolu správnosti vzorkování. Proměnná pro uložení maximálního zpoždění uložení vzorku $samplingDelayMax$, proměnná udávající sumu největšího zpoždění vzorkování během celého měření, tedy za všechny periody, $samplingDelaySumMax$ a proměnná udávající sumu všech zpoždění vzorkování během jedné periody $samplingDelaySum$ se vynulují. Vynulují se i proměnné sloužící ke kontrole podobnosti period $ySamplingArrayDiffSum$ a $ySamplingArraySum$.

Následně se určí vzorkovací perioda $samplingPeriodTimeSpan$

$$samplingPeriodTimeSpan = \frac{T_{period}}{800}. \tag{6.14}$$

Funkce $getRTC()$, na níž je založeno vzorkování, je schopna vrátit čas s přesností na desítky milisekund. Abychom poskytli stroji dostatečný čas na vykonání cyklu programu bylo zvoleno

$$samplingPeriodTimeSpan \geq 20. \tag{6.15}$$

Po splnění této podmínky se nedesítkové hodnoty vzorkovací periody v milisekundách zaokrouhlí na nejbližší vyšší celou desítku. Posledním krokem inicializace je uložení aktuálního času do proměnné T_0.

V hlavní fázi pak dochází k přepínání relé a ukládání aktuálních hodnot akční veličiny a regulované veličiny do připravených polí. Pro hodnoty akční veličiny u je připravené pole $u_{SamplingArray}[X, Y]$, kde $X = samplingArrayColumnIndex$ a $Y = samplingArrayIndex$, o rozměru $3 \times 1\,300$ a pro řízenou veličinu y stejně velké i stejně indexované pole $y_{SamplingArray}[X, Y]$. Dokud nejsou splněny podmínky pro opuštění vzorkování, probíhají následující procesy. Při každém cyklu programu je se určuje index pole $samplingArrayIndex$, do kterého se má uložit aktuální hodnota akční a řízené veličiny. K tomu potřebujeme znát dobu, která uplynula od začátku aktuální periody. Načteme tedy aktuální čas a od něho odečteme čas T_0. Získaný časový úsek vydělíme vzorkovací periodou $samplingPeriodTimeSpan$ a zaokrouhlíme dolů na celá čísla. Pokud je vypočtený index $samplingArrayIndex$ větší než předchozí hodnota tohoto indexu uložená do proměnné $samplingArrayIndexOld$, uloží se aktuální hodnoty akční a regulované veličiny do připravených polí. Následně se určí

rozdíl mezi aktuálně uloženou hodnotou regulované veličiny a hodnotami regulované veličiny se stejným indexem vzorku z předchozích dvou period. Součet absolutních hodnot těchto rozdílů se ukládá do proměnné $y_{SamplingArrayDiffSum}$. Zároveň se aktuální hodnota regulované veličiny uloží do proměnné pro sumu y přes jednu periodu $y_{SamplingArraySum}$. Současně se zjistí, jestli doslo k nějakému posunu oproti ideálnímu času vzorkování. V případě, že je tento časový posun dosud největší, uloží se do proměnné $samplingDelayMax$. Všechny časové posuny v jedné periodě se přičítají k proměnné $samplingDelaySum$. Největší z těchto součtů za celé měření se uloží do proměnné $samplingDelaySumMax$.

K sestupné hraně dojde, když

$$y > y_0 + hysteresis_{HighValue} \tag{6.16}$$

a

$$u = u_{HighValue}, \tag{6.17}$$

tehdy se přepne na $u = u_{LowValue}$.

K náběžné hraně dojde, když

$$y < y_0 - hysteresis_{LowValue} \tag{6.18}$$

a

$$u = u_{LowValue}, \tag{6.19}$$

tehdy se přepne na $u = u_{HighValue}$. Náběžná hrana zároveň signalizuje konec periody. Proto musejí být v momentu náběžné hrany vykonány úkony určené na konec periody. Pokud je suma zpoždění při ukládání vzorků $samplingDelaySum$ v minulé periodě větší než největší suma zpoždění v kterékoliv předchozí periodě $samplingDelaySumMax$, přepíše se tato proměnná aktuální sumou zpoždění vzorkování $samplingDelaySum$, a ta se následně vynuluje pro použití v následující periodě. Vynulují se i ostatní proměnné, které jsou ukazateli přesnosti vzorkování a vztahují se pouze k jedné periodě. Při náběžné hraně se také kontroluje, zda jsou tři sousední periody dostatečně podobné, aby se vzorkování ukončilo. Zvolené kritérium vyžaduje, aby součet procentuálních rozdílů mezi aktuální periodou a dvěma předchozími nepřekročil 6 %. Podmínka pro opuštění vzorkování je tedy

$$\frac{y_{samplingArrayDiffSum}}{y_{samplingArraySum}} < 0.06. \tag{6.20}$$

6.4 Zpracování naměřených dat

6.4.1 CASE 90 a CASE 91 – výpočet integrálů ze vstupu a výstupu soustavy ve frekvenční oblasti

V inicializační fázi je třeba vypočíst periodu kmitu soustavy při vzorkování

$$T_{period} = T_1 + T_2. \tag{6.21}$$

Z periody kmitu při vzorkování soustavy se vypočítá úhlová frekvence

$$\omega_1 = \frac{2\pi}{T_{period}} \tag{6.22}$$

a teoretická frekvence

$$\omega_2 = 2 \cdot \omega_1. \tag{6.23}$$

Pro výpočet integrálu z akční veličiny a regulované veličiny při frekvenci odpovídající měření je třeba vynulovat index ukazující na správné místo v poli uložených hodnot

$$indexIntegral = 0. \tag{6.24}$$

Dále si zavedeme index $indexShiftIntegral$ ukazující na hodnotu akční a regulované veličiny, která je o půl periody posunutá oproti hodnotě s indexem $indexIntegral$

$$indexShiftIntegral = \frac{curentLastIndex}{2} \tag{6.25}$$

a

$$curentLastIndex = samplingArrayLastIndexes[X], \tag{6.26}$$

kde

$$X = samplingArrayColumnIndex. \tag{6.27}$$

Dále se nastaví soustava do neutrálního stavu nastavením hodnoty akční veličiny na $u_{LowValue}$.

V hlavní fázi dochází k samotnému výpočtu integrálů. Při každém cyklu PLC se k integrálu přičtou hodnoty uložené v poli s indexem větším o 1. Když výpočet pokryje všechny prvky aktuálního sloupce pole, je integrace ukončena. Vzhledem k tomu, že vypočítané integrály jsou následně použity ve vztahu, kde je podíl těchto dvou integrálů, které mají stejnou vzorkovací periodou, je možné vzorce pro integraci zjednodušit. Zjednodušení spočívá ve vynechání násobení vzorkovací periodou při numerické integraci, protože v podílu integrálů by stejně došlo k jeho vykrácení. Integraci dle vztahů čitatelů a jmenovatelů ve vztazích (3.16) a (3.20) provádíme postupně. Nejprve si připravíme exponenty pro komplexní exponenciální funkce

$$integralExponenent1 = \omega_1 \cdot tauIndexIntegral \tag{6.28}$$

a

$$integralExponenent2 = \omega_2 \cdot tauIndexIntegral, \tag{6.29}$$

kde

$$tauIndexIntegral = \frac{indexIntegral \cdot samplingPeriodTimeSpan}{1000} \tag{6.30}$$

udává čas vzorku od počátku periody. Takto vypočtené indexy je třeba převést na komplexní číslo, protože při výpočtu integrálu budou použity funkce pro počítání s komplexními čísly [16]. Do komplexní proměnné $u_{sampleComplex1}$ uložíme hodnotu akční veličiny odpovídající času $tauIndexIntegral$. Stejně uložíme do proměnné $y_{sampleComplex1}$ hodnotu regulované veličiny odpovídající času $tauIndexIntegral$. Do komplexní proměnné $u_{sampleComplex2}$ uložíme hodnotu akční veličiny odpovídající

času $tauIndexIntegral$ sečtenou s hodnotou odpovídající času posunutému o půl periody T_{period}. Stejně uložíme do proměnné $y_{sampleComplex2}$ hodnotu regulované veličiny odpovídající času $tauIndexIntegral$ sečtenou s hodnotou regulované veličiny posunuté o půl periody. Takto připravené proměnné se vloží do funkcí $CEXP$, $CMUL$ a $CADD$ z knihovny $OSCATLib$ [16] a vypočítáme hodnoty integrálů $integral_{U1}$, $integral_{Y1}$, $integral_{U2}$ a $integral_{Y2}$. Na konci každého cyklu se proměnná $indexIntegral$ navýší o jedna a proměnná $indexShiftIntegral$ se přepočte na hodnotu posunutou oproti $indexIntegral$ o půl periody T_{period}.

6.4.2 CASE 100 a CASE 101 – výpočet bodů Nyquistovy frekvenční charakteristiky

Inicializační fáze je zde pouze pro zachování přehlednosti skriptu.

V hlavní fázi se vydělením integrálů vypočtených v předchozí části programu vypočítají body Nyquistovy frekvenční charakteristiky odpovídající úhlovým frekvencím ω_1 a ω_2.

$$G_1 = \frac{integral_{Y1}}{integral_{U1}} \tag{6.31}$$

$$G_2 = \frac{integral_{Y2}}{integral_{U2}} \tag{6.32}$$

K dělení komplexních čísel byla použita funkce $CDIV$ volně dostupné knihovny OSCAT Basic library [16]. Její správné fungování bylo ověřeno výpočtem na základě převedení podílu komplexních čísel na zlomek s reálným jmenovatelem. Obě metody dávaly stejné výsledky. Funkce $CDIV$ tedy pracuje korektně.

6.4.3 CASE 110 a CASE 111 – výpočet parametrů SOTD modelu

Inicializační fáze je zde pouze pro zachování zvolené struktury skriptu.

V hlavní fázi je pak pro výpočet parametrů modelu druhého řádu s dopravním zpožděním umístěn pro tuto práci napsaný funkční blok, který z určených parametrů soustavy (G_1, G_2, ω_1, ω_2 a K) vypočítá parametry modelu dosazením do vztahů

$$sotd_{A2} = \frac{1}{\omega_1^2}\sqrt{\frac{1}{3}\left(\frac{K^2}{|G_2|^2} - \frac{4K^2}{|G_1|^2} + 3\right)}, \tag{6.33}$$

$$sotd_{A1} = \frac{1}{\omega_1}\sqrt{\frac{K^2}{|G_1|^2} - \left(1 - sotd_{A2} - \omega_1^2\right)^2}, \tag{6.34}$$

$$sotd_K = K, \tag{6.35}$$

$$sotd_\tau = \frac{1}{2}\left(\sum_{l=1}^{2} \frac{arg\frac{1}{sotd_{A2}(j\omega_l)^2 + sotd_{A1}(j\omega_l) + 1} - arg\left(G_l\right)}{\omega_l}\right). \tag{6.36}$$

Při výpočtech byly použity funkce $CARG$, $CDIV$ a $CABS$ z knihovny OSCAT Basic library pro výpočet argumentu komplexního čísla, dělení komplexních čísel

a výpočet velikosti komplexního čísla [16]. Vypočtenými parametry je dán SOTD model soustavy

$$G(s) = \frac{sotd_K \cdot e^{sotd_\tau \cdot s}}{sotd_{A2} \cdot s^2 + sotd_{A1} \cdot s + 1}. \tag{6.37}$$

6.5 Ukončení měření

6.5.1 CASE 120 a CASE 121 – příprava programu k další identifikaci

Inicializační fáze je zde jen pro zachování struktury projektu.

V hlavní fázi se uživateli vypíše ve webovém rozraní hláška „Identifikace soustavy byla dokončena“. Následně se pro příští průchod programu nastaví ukazatel na $CASE$ 00 a aktivuje se tlačítko *Start* pro další použití programu.

Kapitola 7

Určení parametrů modelu ze tří bodů Nyquistovy frekvenční charakteristiky s přidaným dopravním zpožděním

Skript pro reléovou identifikaci s přidaným dopravním zpožděním vychází ze skriptu popsaného v předchozí kapitole. Proto se zde zaměříme pouze na odlišnosti oproti výše popsanému skriptu.

7.0.1 CASE 70 a CASE 71 – hledání ustálené periody kmitu

Inicializační fáze je zachována v původní formě.

Hlavní fáze je oproti původní verzi delší o výpočet velikosti dopravního zpoždění $trafficDelay$ použitého pro další kmitání relé

$$trafficDelay = \frac{T_{period}}{x}, \tag{7.1}$$

kde x je koeficient pro volbu velikosti dopravního zpoždění v poměru k periodě kmitu bez dopravního zpoždění.

7.1 CASE 80 a CASE 81 – vzorkování vstupu a výstupu soustavy

Inicializační fáze je zachována v původní formě.

Hlavní fáze má v sobě oproti jednodušší verzi přidáno dopravní zpoždění před spádovou a náběžnou hranu. Při detekci dosažení meze hystereze se relé nepřepne, ale vyčká dokud nedoběhne časovač $TimerForDelay$. Tím se prodlouží perioda kmitu T_{period} a sníží frekvence ω_1 a ω_2. To má za následek posun bodů Nyquistovy frekvenční charakteristiky směrem k vyšším kvadrantům.

Kapitola 8

Určení modelu pro nastavení parametrů PI, PD a PID regulátorů

Stejně jako u předchozích dvou skriptů bylo pro programování zvoleno schéma s použitím struktury CASE OF. Nejprve proběhne fáze inicializační, při které se připraví všechny potřebné proměnné. Následuje hlavní fáze, ve které dochází k samotné identifikaci. Probíhají zde měření a výpočty. Tyto dvě fáze jsou na sebe úzce navázány a není možné použít pouze jednu z nich.

8.1 Spuštění identifikačního procesu

8.1.1 CASE 00 a CASE 01 – kontrola zadaných údajů a spuštění identifikace

Inicializační fáze (CASE 00) zde figuruje pouze pro zachování zvoleného schématu a nemá další funkci.

Hlavní fáze (CASE 01) slouží pro zadání parametrů identifikace. Pro zadání parametrů identifikace bylo vytvořeno webové rozhranní (obr. 8.1). Hodnoty se předají do funkčního bloku, který kontroluje jejich správné zadání. Je třeba zadat hodnotu akční veličiny v pracovním bodě, maximální přípustnou hodnotu akční veličiny a minimální přípustnou hodnotu akční veličiny. Po stisknutí tlačítka *Start* se hodnota proměnné $button_{IsPressed}$ nastaví na hodnotu $TRUE$.

Obrázek 8.1: Webové rozhraní pro ovládání programu reléové identifikace.

Zadané parametry identifikace se předají do funkčního bloku *fbwebinputvalidation* napsaného pro tuto práci. Při správném zadání parametrů identifikace se nastaví návratová proměnná *tempBool* na hodnotu $TRUE$. Tlačítko pak zůstane zmáčknuté a přejde se do další fáze identifikace. Pokud jsou parametry zadány chybně, proměnná *tempBool* má hodnotu $FALSE$ a tlačítko se uvolní k dalšímu stisknutí. Podle chyby v zadání parametrů se vypíše chybové hlášení, aby uživatel věděl, jak zadané parametry upravit (tab. 8.1).

chyba	chybová hláška
$u_{MinValue} > u_{MaxValue}$	horní mez musí ležet nad dolní mezí
$u_{MinValue} < 0$	horní a dolní mez musí ležet mezi 0 a 10
$u_{MaxValue} > 10$	horní a dolní mez musí ležet mezi 0 a 10
$(u_0 - 0,5) < u_{MinValue}$	vstup musí ležet mezi (dolní mezí + 0,5) a horní mezí
$u_0 > u_{MaxValue}$	vstup musí ležet mezi (dolní mezí + 0,5) a horní mezí

Tabulka 8.1: Chybové hlášky při špatném zadání parametrů regulace.

8.2 Určení pracovního bodu, statické citlivosti soustavy a velikosti šumu výstupního signálu

8.2.1 CASE 10 a CASE 11 – nastavení konstantní hodnoty vstupu a čekání na ustálený stav

V inicializační fázi (CASE 10) se nastaví hodnota akční veličiny na

$$u = u_0 - 0,5 \tag{8.1}$$

a přejde se do hlavní fáze.

V hlavní fázi (CASE 11) se volá funkční blok *fbstabilization* kontrolující ustálení regulované veličiny y. Jde o stejný funkční blok, jaký byl použit v obou skriptech pro reléovou identifikaci ze tří bodů Nyquistovy frekvenční charakteristiky. Kontrola ustálení regulované veličiny je založena na průměrování hodnoty regulované veličiny po definovaý čas. Následně se poslední tři takto určené průměry porovnají a pokud jejich průběh změnil svojí monotónnost, je soustava považována za ustálenou.

8.2.2 CASE 20 a CASE 21 – určení bodu blízkého pracovnímu bodu

V inicializační fázi (CASE 20) se vynulují proměnné pro použití v hlavní fázi. Do proměnné *tempCounter* se bude ukládat počet vzorků regulované veličiny y a do proměnné *tempSum* jejich součet.

V hlavní fázi se spustí časovač s dobou trvání 1 s. Dokud časovač běží, přičítají se aktuální hodnoty regulované veličiny k proměnné *tempSum*. Proměnná *tempCounter* počítá počet cyklů. Po doběhnutí časovače se vypočítá průměrná hodnota regulované veličiny

$$y_{K1} = \frac{tempSum}{tempCounter} \tag{8.2}$$

v bodě blízkém pracovnímu bodu danému hodnotou akční veličiny

$$u_{K1} = u_0 - 0,5. \tag{8.3}$$

8.2.3 CASE 30 a CASE 31 – nastavení hodnoty vstupu do pracovního bodu a čekání na ustálený stav

V inicializační fázi (CASE 30) se nastaví hodnota akční veličiny na $u = u_0$. Tedy na hodnotu akční veličiny v pracovním bodě.

V hlavní fázi (CASE 31) se znovu čeká na ustálení soustavy. Ustálení je kontrolováno pomocí funkčního bloku *fbstabilization*. Po ustálení se přejde do další fáze.

8.2.4 CASE 40 a CASE 41 – určení pracovního bodu a statické citlivosti soustavy

V inicializační fázi (CASE 40) se vynulují proměnné pro použití v hlavní fázi. Jde o proměnné *tempSum* a *tempCounter*, které v hlavní fázi poslouží stejně jako ve fázi CASE 21.

Průběh hlavní fáze (CASE 41) je z části stejný jako ve fázi CASE 21. Po dobu 1 s měřenou časovačem se počítá suma vzorků regulované veličiny y a ukládá se do proměnné *tempSum*. Počet vzorků je pak uložen v *tempCounter*. Po doběhnutí časovače se vypočte průměrná hodnota akční veličiny v pracovním bodě

$$y_0 = y_{K2} = \frac{tempSum}{tempCounter}. \tag{8.4}$$

Platí $u_{K2} = u_0$. Z takto určených bodů statické charakteristiky se následně vypočítá statická citlivost soustavy

$$K = \frac{y_{K2} - y_{K1}}{u_{K2} - u_{K1}}. \tag{8.5}$$

8.2.5 CASE 50 a CASE 51 – určení maximální hodnoty šumu v pracovním bodě

Jediným úkolem inicializační fáze (CASE 50) je vynulovat proměnou $noise_{Max}$ pro uložení velikosti šumu.

V hlavní fázi (CASE 51) se spustí na 10 s časovač. Po dobu jeho chodu se do proměnné $noise_{Max}$ ukládá maximální výchylka regulované veličiny y od její střední hodnoty v pracovním bodě.

$$noise_{Max} = |y_0 - y| \tag{8.6}$$

Po doběhnutí časovače se z velikosti šumu se vypočte velikost hystereze relé. Pro horní i dolní mez hystereze platí

$$hysteresis_{HighValue} = hysteresis_{LowValue} = 10 \cdot noise_{Max}. \tag{8.7}$$

Vypočtou se hodnoty $y_{MaxHighDeviation}$ a $y_{MaxLowDeviation}$, mezi kterými má při reléové identifikaci kmitat regulovaná veličina. Slovem $High$ jsou označené proměnné vztahující se ke kmitům nad hodnotu v pracovním bodě. Slovem Low pak proměnné mající vztah ke kmitům pod hodnotu v pracovním bodě,

$$y_{MaxLowDeviation} = \nu \cdot hysteresis_{LowValue}, \tag{8.8}$$

$$y_{MaxHighDeviation} = 3 \cdot y_{MaxLowDeviation}, \tag{8.9}$$

kde je deklarací dáno $\nu = 5$.

8.3 Určení parametrů relé

8.3.1 CASE 60 a CASE 61 – exponenciální náběh akční veličiny

V inicializační fázi (CASE 60) se uživateli vypíše informační hlášení „Nastavují se hodnoty pro polohy relé“. Následně se do proměnné $start_{ExponentialTime}$ uloží aktuální čas a přejde se do hlavní fáze.

V hlavní fázi (CASE 61) se bude exponenciálně zvětšovat hodnota akční veličiny. Nejprve se do proměnné $currentTime$ uloží aktuální čas. Pak se určí, kolik času uběhlo od počátku exponenciálního růstu

$$exponentialTimeSpan = currentTime - start_Eponential Time. \tag{8.10}$$

Akční veličina má pak hodnotu

$$u = e^{exponentialTimeSpan \cdot exponencialCorrectionCoeficient} - 1 + u_0, \tag{8.11}$$

kde je deklarací proměnných dáno $exponencialCorrectionCeficient = \frac{1}{3}$. Akční veličina takto v každém strojním cyklu narůstá, dokud nedosáhne hodnoty $u_{MaxValue}$ nebo regulovaná veličina nepřekročí hodnotu $y_{MaxHighDeviation}$. Pokud nastane jeden z uvedených vztahů, určí se hodnota akční veličiny při horní a spodní poloze relé. Do pomocné proměnné $u_{ValueTemp}$ se uloží aktuální hodnota akční veličiny. Do druhé pomocné proměnné $u_{ValueTemp2}$ se uloží taková hodnota, aby její odchylka od akční veličiny tvořila 70 % ochylky u proměnné $u_{ValueTemp}$ a byla opačného směru,

$$u_{ValueTemp2} = u_0 - [(u_{ValueTemp} - u_0) \cdot 0,7]. \tag{8.12}$$

Pro horní polohu relé pak platí

$$u_{HighValue} = u_{ValueTemp} \tag{8.13}$$

a pro dolní polohu

$$u_{LowValue} = u_{ValueTemp2} \tag{8.14}$$

za předpokladu, že $u_{LowValue} \geq u_{MinValue}$. Následuje výpočet stupně asymetrie relé

$$\gamma = \frac{max(|u_{HighValue} - u_0|, |u_{LowValue} - u_0|)}{min(|u_{HighValue} - u_0|, |u_{LowValue} - u_0|)}. \tag{8.15}$$

8.3.2 CASE 70 a CASE 71 – kontrola parametrů relé a jejich případná úprava

V inicilizační fázi (CASE 70) se nejprve nastaví hodnota akční veličiny na $u = u_{LowValue}$. Vynuluje se čítač spádových hran $trailingEdgeCounter$, aktuální čas se uloží do dočasné proměnné pro čas spádové hrany $trailingEdgeRtcTemp$ a náběžné hrany $leadingEdgeRtcTemp$. Vypočítá se aktuální hodnota stupně asymetrie rélé γ (8.15).

V hlavní fázi (CASE 71) se kontroluje, zda již proběhly dvě náběžné hrany. Pokud ne, dochází k přepínání relé. Když $u \neq u_{LowValue}$ a $y \geq y_0 + hysteresis_{HighValue}$,

přepne se relé do dolní polohy, tedy $u = u_{LowValue}$. Z dočasných proměnných pro uložení spádové hrany $trailingEdgeRtcTemp$ a náběžné hrany $leadingEdgeRtcTemp$ se vypočítá předběžně doba, kdy je relé v horní poloze z předchozí periody

$$T_{1Temp} = trailingEdgeRtcTemp - leadingEdgeRtcTemp. \tag{8.16}$$

Do proměnné $trailingEdgeRtcTemp$ se uloží aktuální čas a čítač spádových hran připočte aktuální spádovou hranu. Obdobně když $u \neq u_{HighValue}$ a $y \geq y_0 - hysteresis_{LowValue}$, přepne se relé do dolní polohy, tedy $u = u_{HighValue}$. Z dočasných proměnných pro uložení spádové hrany $trailingEdgeRtcTemp$ a náběžné hrany $leadingEdgeRtcTemp$ se vypočítá předběžně doba, kdy je relé v horní poloze, z předchozí periody

$$T_{2Temp} = leadingEdgeRtcTemp - trailingEdgeRtcTemp. \tag{8.17}$$

Do proměnné $leadingEdgeRtcTemp$ se uloží aktuální čas a čítač náběžných hran připočte aktuální náběžnou hranu.

Pokud nepoběhla žádná spádová, ale proběhla jedna náběžná hrana, určí se maximální výchylka regulované veličiny pod pracovní bod

$$y_{DeflectionLow} = max(y_{DeflectionLow}, |y - y_0|). \tag{8.18}$$

Pokud proběhla jedna náběžná i spádová hrana, určí se maximální výchylka regulované veličiny nad pracovní bod

$$y_{DeflectionHigh} = max(y_{DeflectionHigh}, |y - y_0|). \tag{8.19}$$

Při druhé náběžné hraně a první spádové hraně se vrátíme na počátek vyhodnocování parametrů relé přepsáním hodnot čítačů $leadingEdgeCounter = 1$ a $trailingEdgeCounter = 0$. Stanoví se dočasná hodnota ρ_{Temp}.

$$\rho_{Temp} = \frac{max(T_{1Temp}, T_{2Temp})}{min(T_{1Temp}, T_{2Temp})} \tag{8.20}$$

a pracovní velikost dopravního zpoždění

$$\tau = \frac{\gamma - \rho_{Temp}}{(\gamma - 1)(0,35\rho_{Temp} + 0,65)}. \tag{8.21}$$

Pokud $\tau < 0,05$ upraví se parametry relé. Úpravy parametrů relé probíhají jinak, než je popsáno v Automatic Tuning of PID Controllers based on Asymmetric Relay Feedback [9]. Když byly použity doporučené úpravy parametrů relé, často docházelo k situaci, kdy regulovaná veličina nedosáhla meze hystereze relé. To pak setrvalo v jedné poloze a nemohlo přepnout do druhé. Proto byla úprava parametrů napsána jinak, značně složitěji za použití struktury CASE.

V prvním prvku struktury CASE (CASE – 0) se provádí tyto úpravy. Když pro odchylku regulované veličiny nad horní mez hystereze platí

$$y_{DeflectionHigh} - Hysteresis_{HighValue} > 1 \tag{8.22}$$

a zároveň pro odchylku regulované veličiny pod dolní mez hystereze platí

$$y_{DeflectionLow} - Hysteresis_{LowValue} > 1, \tag{8.23}$$

tak se parametry relé přepočtou podle vztahů

$$u_{HighVaulue} = u_0 + (u_{HighVaulue} - u_0) \cdot 0,8 \tag{8.24}$$

$$u_{LowVaulue} = u_0 + (u_{LowVaulue} - u_0) \cdot 0,8. \tag{8.25}$$

Když pro odchylku regulované veličiny nad horní mez hystereze platí

$$y_{DeflectionHigh} - Hysteresis_{HighValue} < 0,1 \tag{8.26}$$

a zároveň pro odchylku regulované veličiny pod dolní mez hystereze platí

$$y_{DeflectionHigh} - Hysteresis_{HighValue} < 0,1, \tag{8.27}$$

tak se nastaví hodnota ukazatele na prvky struktury CASE $yConditions = 1$. Když pro odchylku regulované veličiny nad horní mez hystereze platí

$$y_{DeflectionHigh} - Hysteresis_{HighValue} < 1 \tag{8.28}$$

a zároveň pro odchylku regulované veličiny pod dolní mez hystereze platí

$$y_{DeflectionLow} - Hysteresis_{LowValue} < 1, \tag{8.29}$$

a také pro odchylku regulované veličiny nad horní mez hystereze platí

$$y_{DeflectionHigh} - Hysteresis_{HighValue} > 0,5 \tag{8.30}$$

a současně pro odchylku regulované veličiny pod dolní mez hystereze platí

$$y_{DeflectionLow} - Hysteresis_{LowValue} > 0,5, \tag{8.31}$$

tak se parametry relé přepočtou podle vztahů

$$u_{HighVaulue} = u_0 + (u_{HighVaulue} - u_0) \cdot 0,9 \tag{8.32}$$

$$u_{LowVaulue} = u_0 + (u_{LowVaulue} - u_0) \cdot 0,9. \tag{8.33}$$

Když pro odchylku regulované veličiny nad horní mez hystereze platí

$$y_{DeflectionHigh} - Hysteresis_{HighValue} < 0,5 \tag{8.34}$$

a zároveň pro odchylku regulované veličiny pod dolní mez hystereze platí

$$y_{DeflectionLow} - Hysteresis_{LowValue} < 0,5, \tag{8.35}$$

a také pro odchylku regulované veličiny nad horní mez hystereze platí

$$y_{DeflectionHigh} - Hysteresis_{HighValue} > 0,1 \tag{8.36}$$

a současně pro odchylku regulované veličiny pod dolní mez hystereze platí

$$y_{DeflectionLow} - Hysteresis_{LowValue} > 0,1, \tag{8.37}$$

tak se parametry relé přepočtou podle vztahů

$$u_{HighVaulue} = u_0 + (u_{HighVaulue} - u_0) \cdot 0,95 \tag{8.38}$$

$$u_{LowVaulue} = u_0 + (u_{LowVaulue} - u_0) \cdot 0,95. \tag{8.39}$$

Ve druhém prvku struktury CASE (CASE – 1) se provádí tyto úpravy. Když platí

$$y_{DeflectionHigh} - Hysteresis_{HighValue} > y_{DeflectionLow} - Hysteresis_{LowValue} \tag{8.40}$$

tak se parametry relé přepočtou podle vztahů

$$u_{HighVaulue} = u_0 + (u_{HighVaulue} - u_0) \cdot 0,9 \tag{8.41}$$

Jinak se provede přepočet parametrů

$$u_{LowVaulue} = u_0 + (u_{LowVaulue} - u_0) \cdot 0,9. \tag{8.42}$$

8.4 Měření průběhů akční a regulované veličiny

8.4.1 CASE 80 a CASE 81 – vzorkování výstupu soustavy

Inicializační fáze (CASE 80) slouží k výpočtu stupně asymetrie relé γ definovaného vztahem (5.3) pomocí předpisu (8.15).

V hlavní fázi (CASE 71) se vzorkuje regulovaná veličina a její hodnoty se ukládají do pole $y_{SamplingArray}$. To má 3 sloupce a 3 000 řádků. Ukládají se do něho tři po sobě jdoucí periody. Na řádek pole ukazuje proměnná $samplingArrayIndex$ a na sloupec $samplingArrayColumnIndex$. Použitý způsob vzorkování byl převzat z programu poskytnutého Milanem Bydžovským z firmy Teco a.s.. Při každém průchodu programem se kontroluje, zda není třeba přepnout relé.

Pokud $y_0 - y < -hysleresis_{HighValue}$ a zároveň $u \neq u_{LowValue}$, pak je čas na spádovou hranu. V čase spádové hrany se vypočítá čas, kdy bylo relé v dolní poloze v předchozí periodě.

$$T_2 = leadingEdgeRtc - trailingEdgeRtc, \tag{8.43}$$

kde $leadingEdgeRtc$ je čas náběžné hrany a $trailingEdgeRtc$ je čas spádové hrany. Následně do proměnné $trailingEdgeRtc$ uložíme aktuálního hodnotu času. Hodnota indexu $samplingArrayIndex$, při kterém došlo ke spádové hraně, se uloží do pole $trailingEdgeIndex$. V každém cyklu proběhne několik kontrol a výpočtů. Spočítá se perioda kmitu

$$T_p = T_1 + T_2. \tag{8.44}$$

Vypočítá se čas od počátku periody $TimeInPeriod$, který je dán rozdílem mezi časem náběžní hrany a aktuálním časem. Z takto určeného času se vypočete index pro ukládání vzorků $samplingArrayIndex$

$$samplingArrayIndex = \frac{TimeInPeriod}{samplingPeriodTimeSpan}. \tag{8.45}$$

Pokud nejsou všechny řádky pole zaplněné, připočítá se k odchylkám mezi měřeními $y_{SamplingArrayDiffA}$ a $y_{SamplingArrayDiffA}$ rozdíl mezi vedlejšími sloupci pole a aktuální hodnotou výstupu y. Aktuální hodnota regulované veličiny y se uloží do pole $y_{SamplingArray}$ na řádek s indexem $samplingArrayIndex$. Hodnota tohoto indexu se uloží do proměnné $samplingArrayLastIndex$, která udává hodnotu posledního předchozího indexu.

Pokud $y_0 - y > hysleresis_{LowValue}$ a zároveň $u \neq u_{HighValue}$, pak je čas na náběžnou hranu. Program je postaven tak, že náběžnou hranou končí perioda kmitu a začíná následující. Vypočte se doba, kdy je relé v horní poloze

$$T_1 = trailingEdgeRtc - leadingEdgeRtc, \tag{8.46}$$

a do proměnné $leadingEdgeRtc$ se uloží aktuální čas. Protože končí perioda a začíná nová, přejde se do dalšího sloupce v poli $y_{SamplingArray}$. Průměrné rozdíly mezi naměřenými periodami, tedy proměnné $y_{SamplingArrayDiffA}$ a $y_{SamplingArrayDiffA}$ vydělené hodnotou proměnné $samplingArrayLastIndex$ navýšenou o 1, se uloží do proměnných v pořadí podle sloupce v poli, do kterého jsme právě ukládali $y_{SamplingArrayDiff01}$, $y_{SamplingArrayDiff02}$ a nebo $y_{SamplingArrayDiff12}$. V průběhu měření se vyčíslí všechny tři a spolu tvoří opouštěcí podmínku pro ukončení vzorkování. Při spádové hraně se kontroluje opouštěcí podmínka pro dostatečně ustálené kmity. Když platí, že průměrné rozdíly mezi prvky uloženými během tří period $y_{SamplingArrayDiff01}$, $y_{SamplingArrayDiff02}$ a $y_{SamplingArrayDiff12}$ jsou menší než je nastavená hodnota $deviation$, může se přejít ke zpracování naměřených dat. Pokud tomu tak není, pokračuje se ve vzorkování.

8.5 Zpracování naměřených dat a příprava na další identifikaci

8.5.1 CASE 90 a CASE 91 – výpočet integrálů ze vstupu a výstupu soustavy ve frekvenční oblasti

Inicializační fáze (CASE 90) je nejsložitější inicializační fází celého programu. Nejprve se z počtu vzorků $SamplingArrayLastIndexes$ a hodnoty vzorkovací periody $samplingPeriodTimeSpan$ vypočítá perioda kmitu T_p. Podobně se z čísla vzorku zaznamenaného při poslední spádové hraně $trailingEdgeIndex$ a vzorkovací periody $samplingPeriodTimeSpan$ určí čas, kdy je relé v horní poloze T_1. Čas po který je relé v dolní poloze, je dán rozdílem těchto časů. Normalizované dopravní zpoždění τ je dáno

$$\tau = \frac{\gamma - \rho}{(\gamma - 1)(0,35 \cdot \rho + 0,65)}, \tag{8.47}$$

kde

$$\rho = \frac{max(T_1, T_2)}{min(T_1, T_2)} \tag{8.48}$$

a hodnotu proměnné γ již známe. Pokud platí

$$0,05 \leq \tau \leq 1, \tag{8.49}$$

pak se uživateli zobrazí hlášení „Probíhají výpočty parametrů regulátoru“. Hodnota akční veličiny se nastaví na nejnižší povolenou, tedy $u_0 = u_{MinValue}$ a přejde do následující fáze. Pokud podmínka (8.49) není splněná, je třeba změnit parametry relé. Vhodná úprava je zvolena podle podmínek (8.50) a (8.52). Když platí

$$T_1 - T_2 < 0, \tag{8.50}$$

upraví se hodnota akční veličiny při horní poloze relé

$$u_{HighValue} = u_0 + (u_{HighValue} - u_0) \cdot coef_{Relay}. \tag{8.51}$$

Při splnění druhé podmínky

$$T_1 - T_2 > 0 \tag{8.52}$$

se upraví hodnota akční veličiny v dolní poloze relé

$$u_{LowValue} = u_0 + (u_{LowValue} - u_0) \cdot coef_{Relay}. \tag{8.53}$$

V obou případech se proměnná $coef_{Relay} = 0,9$. Po přepočtení parametrů relé se vypočítá aktuální hodnota proměnné γ, uživateli se zobrazí hlášení „Měření se opakuje, program přepočítává parametry relé“ a běh programu se po vynulování potřebných proměnných vrátí do fáze CASE 80.

Hlavní fáze (CASE 91) slouží k výpočtu parametrů modelu. Vztahu (5.17) odpovídá výpočet

$$divisionT_dT = \frac{\tau}{(1-\tau)}. \tag{8.54}$$

K výpočtu časové konstanty T ze vztahu (5.18) byly zavedeny dvě pomocné proměnné *tempReal* a *tempReal*2 .Proměnná *tempReal* je dána jako součin vzorkovací periody a indexu pole, v němž došlo ke spádové hraně. Udává tedy čas, po který bylo relé v horní poloze. Do proměnné *tempReal* se uloží hodnota čitatele v logaritmu ve jmenovateli zlomku ve vztahu (5.18). Časovou konstantu pak spočítáme jako

$$T = \frac{tempReal}{\ln\left(\frac{tempReal2}{\left|u_{MaxValue-u_0}\right| - \frac{hysteresis_{HighValue}}{|K|}}\right)}. \tag{8.55}$$

Velikost dopravního zpoždění T_d se pak vypočte dle vztahu (5.19) jako

$$T_d = T \cdot divisionT_dT. \tag{8.56}$$

8.5.2 CASE 100 a CASE 101 – příprava na další identifikaci

Inicializační fáze (CASE 100) slouží pouze pro zachování schématu programu.

V hlavní fázi (CASE 101) se uvolní tlačítko *start* pro další použití, vymaže se informační hlášení a program se vrátí do fáze CASE 00 a je připravený na další identifikaci.

Část IV

Testování skriptů

Kapitola 9
Použité soustavy

9.1 Simulovaná soustava

Pro prvotní testování a ověření správné funkce skriptů byla v prostředí Mosaic vytvořena simulovaná soustava. Při její tvorbě byly použity funkční bloky z knihovny ModelLib [17]. Funkční blok $fbFirstOrder$ je simulací soustavy prvního řádu s přenosem

$$\frac{Y(s)}{U(s)} = \frac{G}{T_1 s + 1}, \tag{9.1}$$

kde $Y(s)$ je Laplaceův obraz regulované veličiny, $U(s)$ je Laplaceův obraz akční veličiny, G je statická citlivost a T_1 časová konstanta. Do funkčního bloku se zadává statická citlivost, časová konstanta v sekundách a perioda vzorkování T v sekundách. Aby měla simulovaná soustava stejnou rychlost, jakou by měla reálná soustava se stejným přenosem, je třeba, aby byla nastavena na

$$T = 0,01\ s. \tag{9.2}$$

Na vstup funkčního bloku pak přivedeme akční veličinu u a z výstupu čteme hodnoty regulované veličiny y. Zřetězením více funkčních bloků soustavy prvního řádu je možné vytvořit soustavu libovolného řádu. Tímto způsobem byla pro testování vytvořena soustava třetího řádu s přenosem

$$G(s) = \frac{2}{1s + 1} \cdot \frac{1}{1,5s + 1} \cdot \frac{1}{2s + 1}. \tag{9.3}$$

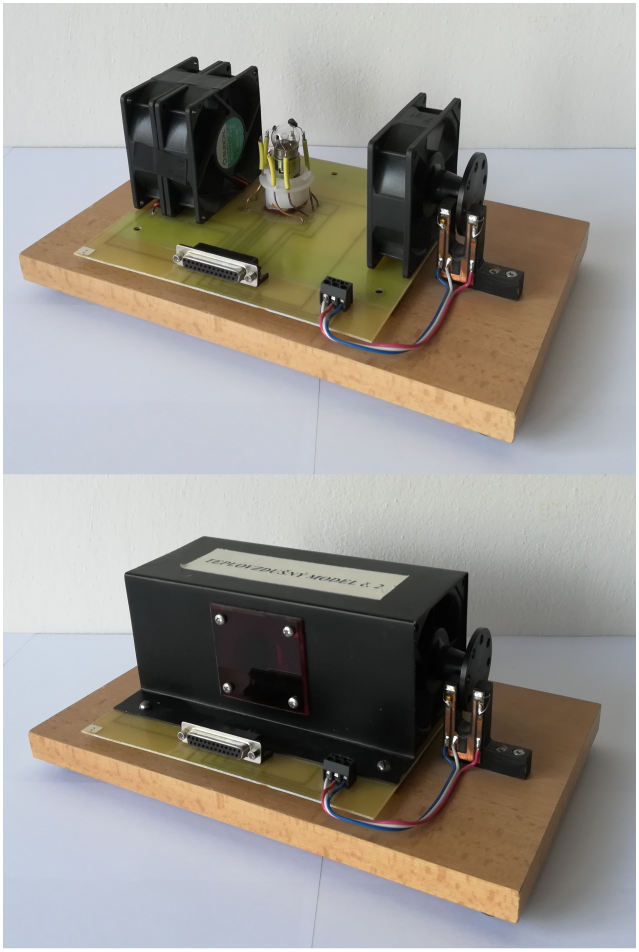
Soustava je stabilní a nekmitavá a je dostatečně vysokého řádu, aby mohla být identifikována FOTD i SOTD modelem.

9.2 Reálné soustavy

Pro odzkoušení napsaných skriptů na reálných soustavách, byla v laboratoři 111 Fakulty strojní ČVUT v Praze, Technická 4 Dejvice vybrána laboratorní soustava „Teplovzdušný model“. Ta byla použita ve dvou různých zapojeních, a je tedy možné ji považovat za dvě různé soustavy.

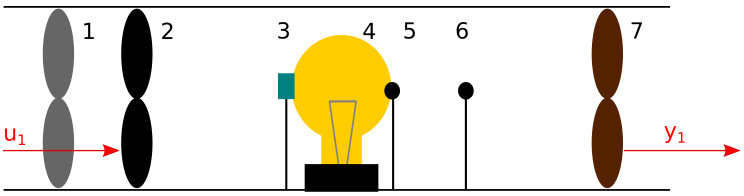
9.2.1 Soustava „Teplovzdušný model“ – zapojení s ventilátorem

Soustava „Teplovzdušný model“ se skládá ze žárovky, dvou ventilátorů, průtokoměru, dvou termistorů a teplotního senzoru KTY82. Vše je umístěno v krytém tunelu. Tři možné akční členy a pět senzorů umožňují připojit soustavu pro více variant měření (obr. 9.1) [18].



Obrázek 9.1: Soustava „Teplovzdušný model“ (Fotografie ve spolupráci s Františkem Hylmarem).

Pro účely této práce bylo použito zapojení s ventilátorem pro měření průtoku vzduchu. Tunel je na jedné straně vybaven dvěma ventilátory. Na vnitřní straně je hlavní ventilátor, na vnější pak poruchový ventilátor, který může pusobit proti hlavnímu ventilátoru. Na druhém konci tunelu se nachází vrtulkový průtokoměr (obr. 9.2).



1 vedlejší poruchový ventilátor
2 hlavní ventilátor
3 teplotní senzor KTY82
4 žárovka
5 termistor
6 termistor
7 vrtulkový průtokoměr

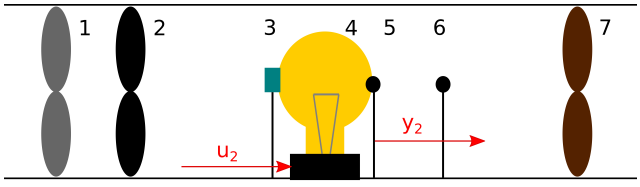
Obrázek 9.2: Schéma soustavy „Teplovzdušný model“ – zapojení s ventilátorem.

Soustava byla zapojena tak, že akční zásah z PLC u_1 ovládá hlavní ventilátor. Výstupem soustavy je tedy průtok vzduchu měřený vrtulkovým průtokoměrem. Průtok vzduchu y_1 jde jako regulovaná veličina na vstup do PLC.

Soustava se chová nelineárně, ale má rychlou odezvu na změnu akční veličiny. Mohlo by se zde však projevit dopravní zpoždění dané délkou tunelu a mechanickými vlastnostmi ventilátoru.

9.2.2 Soustava „Teplovzdušný model“ – zapojení se žárovkou

U laboratorní úlohy „Teplovzdušný model“, která je popsaná v předcházejícím oddíle, proběhla měření i v zapojení se žárovkou. Žárovka je napájená ovládaným zdrojem šířkově modulovaného napětí a je tepelným zdrojem. V těsné blízkosti žárovky je umístěn termistor. Napájení žárovky je řízeno akční veličinou u_2, která je výstupem PLC. Teplota žárovky y_2 snímaná termistorem je regulovanou veličinou a vrací se zpět do PLC (obr. 9.3).



1 vedlejší poruchový ventilátor
2 hlavní ventilátor
3 teplotní senzor KTY82
4 žárovka
5 termistor
6 termistor
7 vrtulkový průtokoměr

Obrázek 9.3: Schéma soustavy „Teplovzdušný model“ – zapojení se žárovkou.

Soustava má pomalou odezvu na změnu akční veličiny. Chová se nelineárně a rychleji se zahřívá, než chladne.

Kapitola 10

Určení parametrů modelu ze tří bodů Nyquistovy frekvenční charakteristiky

10.1 Testování na simulované soustavě

10.1.1 Opakovatelnost měření

K ověření správné funkce napsaného skriptu reléové identikace ze tří bodů frekvenční charakteristiky bylo opakovaně provedeno měření na simulované soustavě (9.3). Měření bylo opakováno desetkrát ve stejném pracovním bodě $u_0 = 3$ při stejně nastavených parametrech identifikace.

	T_p	ω_1	T_{vz}	$G(j\omega_1)$	$G(j\omega_2)$	K
1	8,420	0,746222	20	-0,463587-0,370820i	-0,140208+0,0338541i	2
2	8,420	0,746222	20	-0,462705-0,371940i	-0,140379+0,0331991i	2
3	8,430	0,745336	20	-0,463576-0,374133i	-0,132614+0,0278985i	2
4	8,410	0,747109	20	-0,462159-0,374703i	-0,133510+0,0235349i	2
5	8,420	0,746222	20	-0,462164-0,377968i	-0,125953+0,0176973i	2
6	8,420	0,746222	20	-0,463264-0,371154i	-0,140388+0,0337854i	2
7	8,420	0,746222	20	-0,462130-0,378016i	-0,125901+0,0175906i	2
8	8,430	0,745336	20	-0,463576-0,374133i	-0,132614+0,0278985i	2
9	8,420	0,746222	20	-0,463530-0,370887i	-0,140264+0,0337614i	2
10	8,420	0,746222	20	-0,463432-0,370942i	-0,140263+0,0336730i	2

Tabulka 10.1: Určené body Nyquistovy frekvenční charakteristiky a parametry měření pro 10 různých měření. T_p je perioda kmitu v s, ω_1 je frekvence v rad/s a T_{vz} je vzorkovací perioda v ms. Tři určené body Nyquistovy frekvenční charakteristiky jsou K, $G(j\omega_1)$ a $G(j\omega_2)$.

Skutečnost, že se periody kmitu liší o desítky milisekund, logicky vyplývá z toho, že funkce *getRTC* vrací čas v desítkách milisekund. Perioda kmitu se vypočítá jako rozdíl dvou takto získaných časů. Podle toho, k jaké odchylce dojde při určení

času, je dána přesnost určení periody kmitu T_p. Statická citlivost soustavy K byla určena vždy naprosto přesně. Odlišnost mezi určenými body $G(j\omega_1)$ a $G(j\omega_2)$ je dána rozdílem v naměřené periodě kmitu. U reálné složky bodu $G(j\omega_1)$ činí tento rozdíl $0,32$ % a u imaginární složky $1,9$ %. Pro reálnou složku bodu $G(j\omega_2)$ je rozdíl mezi měřeními 10 % a u imaginární složky 53 %. Pohybujeme se však v desetinách a setinách, a tak se každá odchylka, která vzniká při sčítání dvou hodnot průběhů, projeví obzvlášť výrazně.

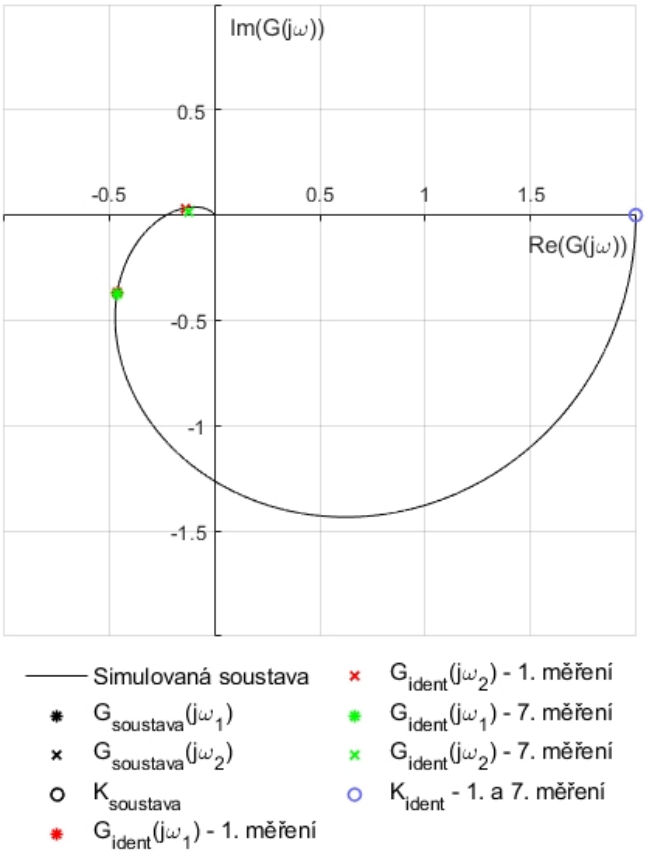
Pro určené body frekvenční charakteristiky byly vypočteny parametry modelu SOTD.

	a_1	a_2	T_d	K
1	2,97518	6,34641	-1,88640	2
2	2,97569	6,34563	-1,88943	2
3	2,43042	6,88836	-2,02796	2
4	2,44131	6,85029	-2,03261	2
5	1,57485	7,42926	-2,22712	2
6	2,98342	6,33729	-1,88515	2
7	1,63844	7,43394	-2,22928	2
8	2,43040	6,88836	-2,02796	2
9	2,97681	6,34453	-1,88633	2
10	2,97637	6,34561	-1,88681	2

Tabulka 10.2: Určené parametry modelu pro 10 různých měření.

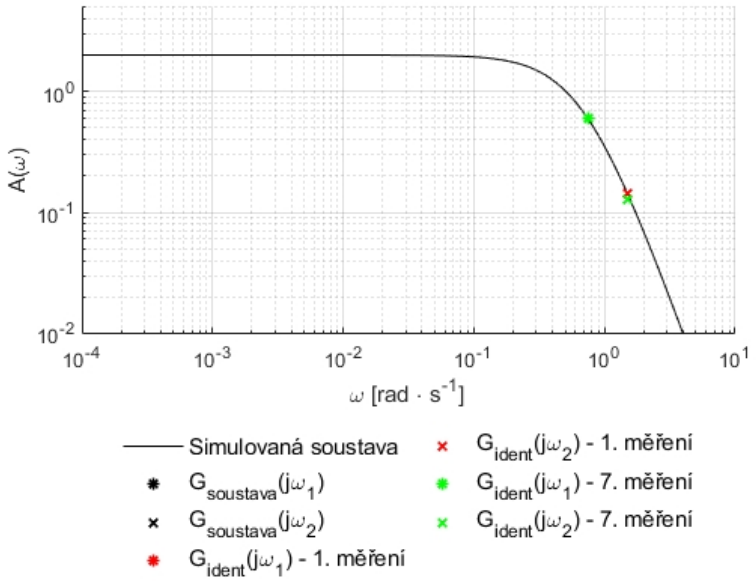
Právě výše uvedená skutečnost, že imaginární složka druhého bodu je malá, má za následek, že pro žádné měření nebyly správně určeny parametry modelu. Dopravní zpoždění totiž ve všech případech vyšlo záporné.

10.1.2 Správnost identifikace simulované soustavy



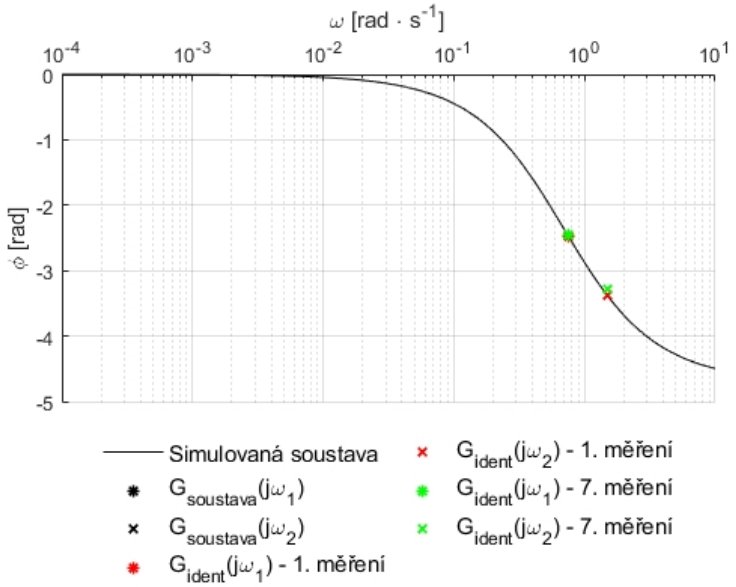
Obrázek 10.1: Nyquistovy frekvenční charakteristiky pro dva nejodlišnější výsledky měření, $\omega_1 = 0,7462\ rad/s$ a $\omega_2 = 1,4924\ rad/s$.

Na Nyquistových charakteristikách je vidět, že body určené identifikací jsou téměř totožné (obr. 10.1). Jen body $G(j\omega_2)$ se liší. Větší odchylka než u bodu $G(j\omega_1)$ je dána způsobem výpočtu ze vztahu (3.20).



Obrázek 10.2: Amplitudové frekvenční charakteristiky pro dva nejodlišnější výsledky měření, $\omega_1 = 0,7462\ rad/s$ a $\omega_2 = 1,4924\ rad/s$.

Podobná odchylka, jaká byla patrná u polohy bodu $G(j\omega_2)$ na Nyquistově frekvenční charakteristice, je i na amplitudové frekvenční charakteristice (obr. 10.2).



Obrázek 10.3: Fázové frekvenční charakteristiky pro dva nejodlišnější výsledky měření, $\omega_1 = 0,7462\ rad/s$ a $\omega_2 = 1,4924\ rad/s$.

Také na fázové frekvenční charakteristice je bod $G(j\omega_2)$ mírně odchýlen (obr. 10.3). Avšak i přes tuto odchylku by mohla být soustava správně identifikována. Statická

citlivost byla ve všech případech určena přesně. Bod $G(j\omega_1)$ z identifikace odpovídá bodu frekvenční charakteristiky soustavy se stejnou frekvencí. Bod $G(j\omega_2)$ z identifikace je oproti bodu frekvenční charakteristiky soustavy mírně posunut, má mírně větší fázi a menší amplitudu. Výraznější problém je však ve výpočtu parametrů modelu, který ve všech případech selhal.

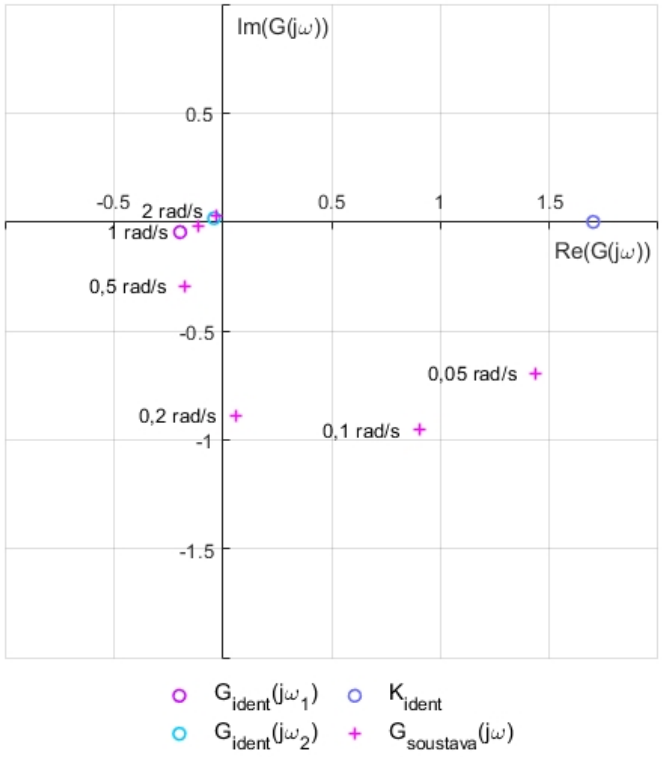
10.2 Testování na reálných soustavách

10.2.1 Identifikace soustavy „Teplovzdušný model“ – zapojení s ventilátorem

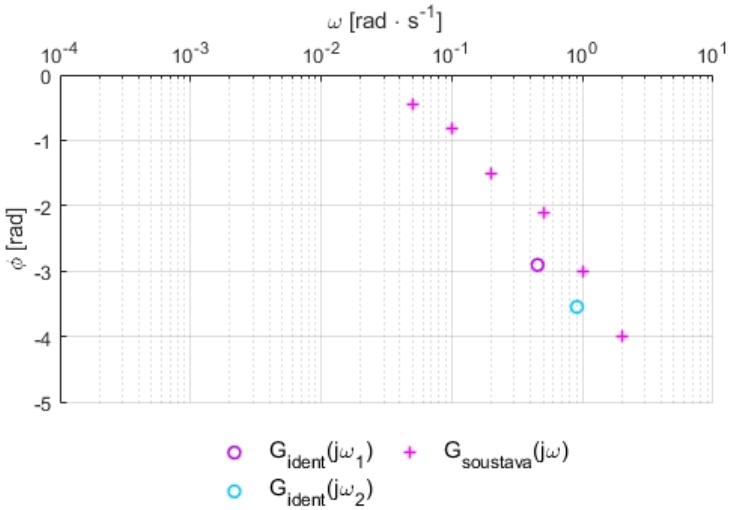
Pro porovnání určeného modelu a reálné soustavy byly naměřeny frekvenční charakteristiky v pracovním bodě $u_0 = 2,3\ V$. Program pro reléovou identifikaci určil body uvedené v tabulce 10.3.

K	$\omega_1\ [rad/s]$	$G(j\omega_1)$	$\omega_2\ [rad/s]$	$G(j\omega_2)$
1.702	0,4527	-0,195681-0,0469211i	0,9054	-0,0394694+0,0169869i

Tabulka 10.3: Reléovou identifikací určené body frekvenčních charakteristik.

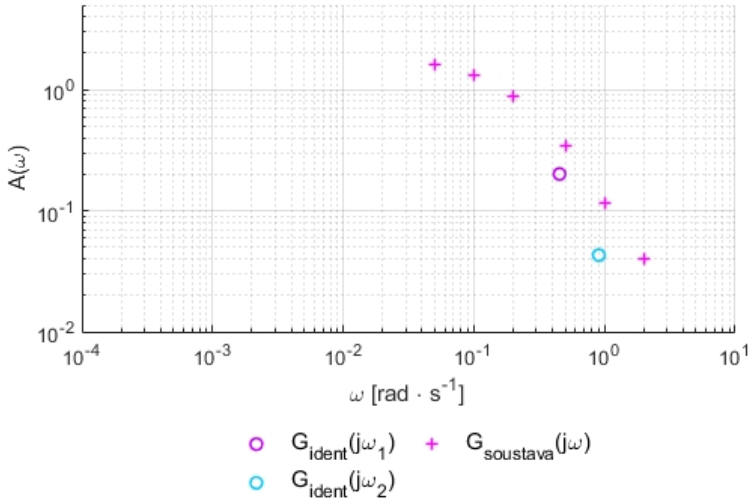


Obrázek 10.4: Porovnání bodů frekvenční charakteristiky určených identifikací s body Nyquistovy frekvenční charakteristikou soustavy.



Obrázek 10.5: Porovnání bodů frekvenční charakteristiky určených identifikací s body fázové frekvenční charakteristikou soustavy.

Naměřené body mají nižší fázi než jim odpovídající body naměřené frekvenční charakteristiky. Amplituda určených bodů je také nižší, než se očekává podle naměřených bodů frekvenční charakteristiky soustavy.



Obrázek 10.6: Porovnání bodů frekvenční charakteristiky určených identifikací s body amplitudové frekvenční charakteristikou soustavy.

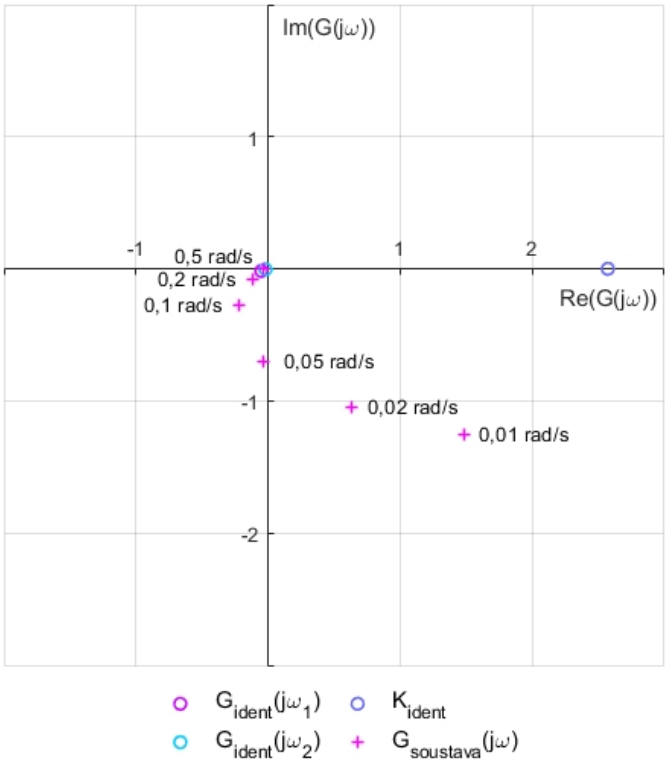
10.2.2 Identifikace soustavy „Teplovzdušný model“ – zapojení se žárovkou

Pro porovnání určeného modelu a reálné soustavy byly naměřeny frekvenční charakteristiky v pracovním bodě $u_0 = 3\ V$. Program pro reléovou identifikaci určil body

uvedené v tabulce 10.4.

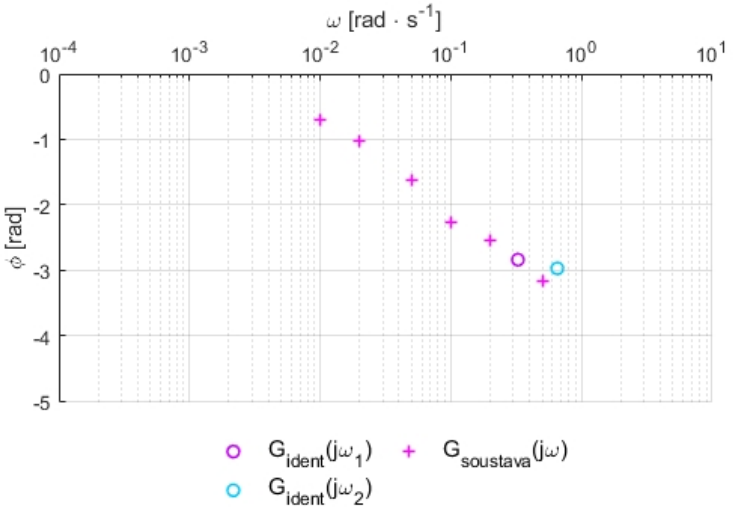
K	ω_1 $[rad/s]$	$G(j\omega_1)$	ω_2 $[rad/s]$	$G(j\omega_2)$
2,57231	0,3261	-0,0509567-0,0158128i	0,6522	-0,0193065-0,00327611i

Tabulka 10.4: Reléovou identifikací určené body frekvenčních charakteristik.

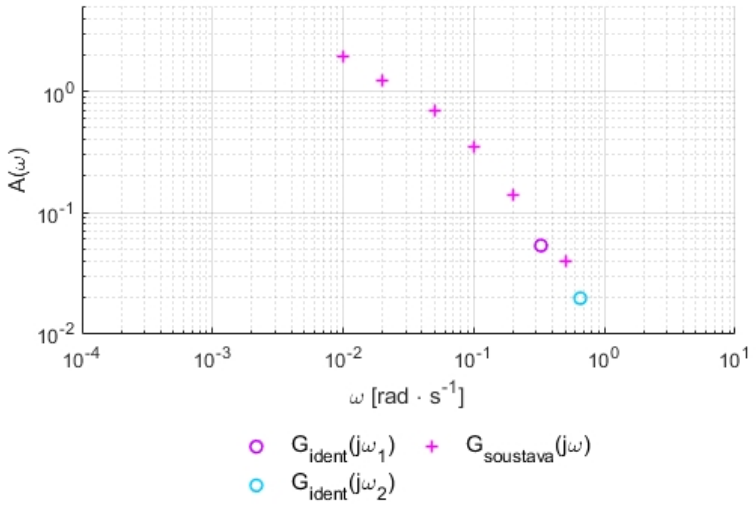


Obrázek 10.7: Porovnání bodů frekvenční charakteristiky určených identifikací s body Nyquistovy frekvenční charakteristiky soustavy.

Bod o frekvenci identifikace a o dvojnásobné frekvenci se na Nyquistově frekvenční charakteristice téměř překrývají (obr. 10.7). Z toho vyplývá, že určené body se nacházejí na části Nyquistovy frekvenční charakteristiky, kde frekvence narůstá rychle a amplituda spěje k nule.



Obrázek 10.8: Porovnání bodů frekvenční charakteristiky určených identifikací s body fázové frekvenční charakteristiky soustavy.



Obrázek 10.9: Porovnání bodů frekvenční charakteristiky určených identifikací s body amplitudové frekvenční charakteristiky soustavy.

Body fázové frekvenční charakteristiky z identifikace vyznačené u bodů frekvenční charakteristiky soustavy ukazují, že fáze druhého určeného bodu neodpovídá zcela očekávanému průběhu frekvenční charakteristiky (obr. 10.8).

Porovnání bodů z identifikace v amplitudové frekvenční charakteristice soustavy ukazuje na dobře určenou amplitudu bodů (obr. 10.9). Na přesnost určení bodů frekvenční charakteristiky má při vysokých frekvencích a malých amplitudách výrazný vliv šum. Bylo by tedy dobré provádět měření při nižších frekvencích přepínání relé.

Kapitola 11

Určení parametrů modelu ze tří bodů Nyquistovy frekvenční charakteristiky s přidaným dopravním zpožděním

11.1 Testování na simulované soustavě

11.1.1 Opakovatelnost měření

Pro kontrolu, zda je program správně napsán, bylo provedeno deset měření na simulované soustavě v pracovním bodě daném hodnotou akční veličiny $u_0 = 3$. Za stejných podmínek měření byla naměřena mírně odlišná data (tab. 11.1).

	T_p	ω_1	T_{vz}	$G(j\omega_1)$	$G(j\omega_2)$	K
1	13,380	0,469595	20	-0,249839-1,04990i	-0,352255-0,148070i	2
2	13,390	0,469245	20	-0,250929-1,04982i	-0,362494-0,142203i	2
3	13,380	0,469595	20	-0,249839-1,04990i	-0,352255-0,148070i	2
4	13,380	0,469595	20	-0,249839-1,04990i	-0,352255-0,148070i	2
5	13,390	0,469245	20	-0,253002-1,05006i	-0,360224-0,140610i	2
6	13,380	0,469595	20	-0,249839-1,04990i	-0,352255-0,148070i	2
7	13,390	0,469245	20	-0,253762-1,04924i	-0,363287-0,141333i	2
8	13,410	0,468545	20	-0,249080-1,05108i	-0,366488-0,144561i	2
9	13,380	0,469595	20	-0,251083-1,04865i	-0,371078-0,139313i	2
10	13,380	0,469595	20	-0,251376-1,049490	-0,352395-0,146854i	2

Tabulka 11.1: Určené body Nyquistovy frekvenční charakteristiky a parametry měření pro 10 různých měření. T_p je perioda kmitu v s, ω_1 je frekvence v rad/s a T_{vz} je vzorkovací perioda v ms. Tři určené body Nyquistovy frekvenční charakteristiky jsou K, $G(j\omega_1)$ a $G(j\omega_2)$.

Stejně jako u předchozí verze reléové identifikace ze tří bodů Nyquistovy frekvenční charakteristiky byly zaznamenány rozdíly v naměřené periodě kmitu. To je způsobeno

skutečností, že funkce *GetRTC*, která byla v průběhu celého programu používána, vrací hodnotu času v desítkách milisekund. Nejmenší možné vzorkování je nastaveno na 20 *ms*. Uvážíme-li se oba dva tyto aspekty, je zřejmé, že odchylky v hodnotách periody kmitu T_p nejsou dány chybou v programu, ale možnostmi PLC. Z periody kmitu se počítá frekvence ω_1 a ta pak poslouží v exponenciální funkci při výpočtu bodů Nyquistovy frekvenční charakteristiky pomocí vztahů (3.16) a (3.20). Rozdíl v určené periodě se tak přenáší až na nalezené body, a tedy i na parametry modelu (tab. 11.2).

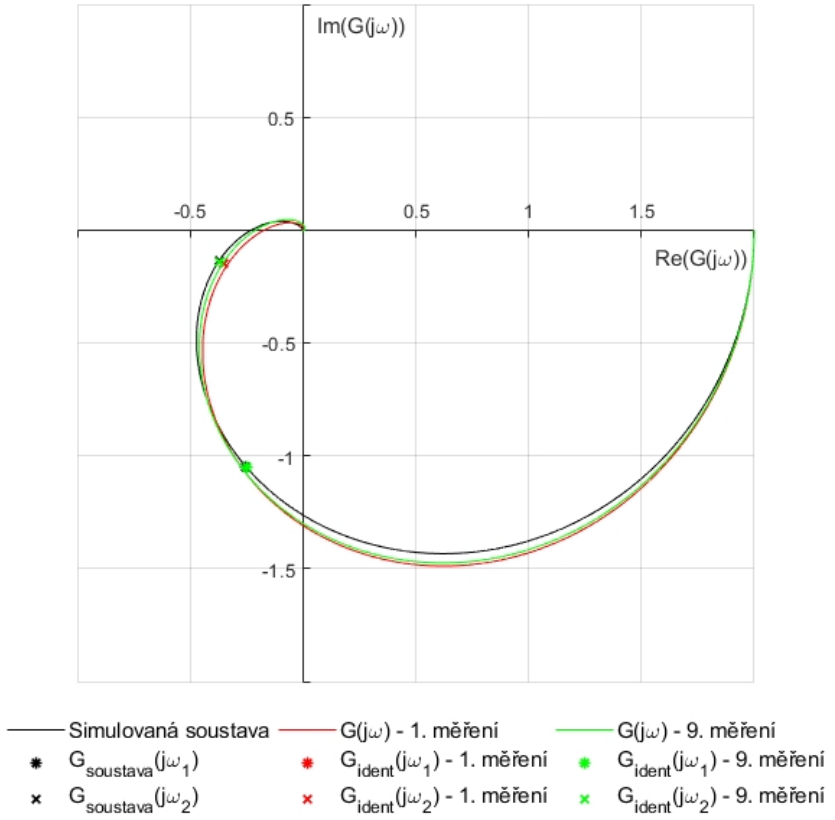
	a_1	a_2	T_d	K
1	3,92808	5,34293	0,350768	2
2	3,93708	5,18613	0,396884	2
3	3,92808	5,34293	0,350768	2
4	3,92808	5,34293	0,350768	2
5	3,93203	5,24988	0,387062	2
6	3,92762	5,35293	0,350768	2
7	3,93703	5,17707	0,402977	2
8	3,94331	5,10272	0,413492	2
9	3,94319	5,01895	0,436440	2
10	3,92644	5,35021	0,352630	2

Tabulka 11.2: Určené parametry modelu pro 10 různých měření.

11.1.2 Správnost identifikace simulované soustavy

Z různých výsledků měření zaznamenaných v tabulkách 11.1 a 11.2 byla vybrána dvě nejrozdílnější měření, konkrétně měření č.1 a měření č.9, a jejich výsledky byly zobrazeny ve frekvenčních charakteristikách.

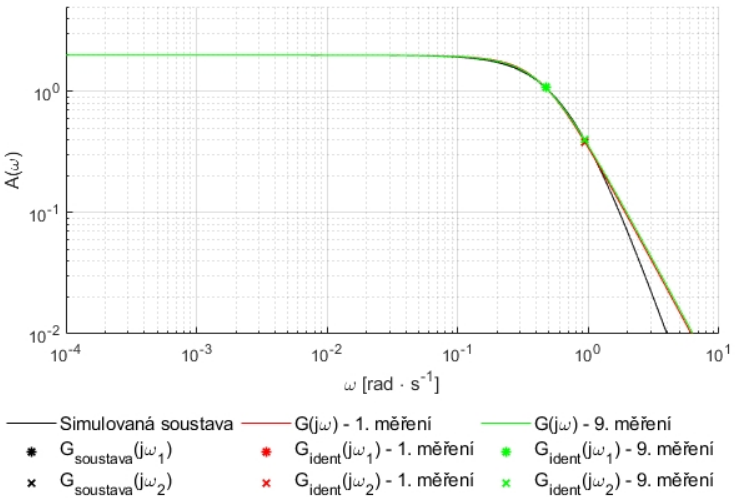
Nyquistovy frekvenční charakteristiky modelů jsou velmi podobné (obr. 11.1). Je však patrné, že ve třetím kvadrantu lépe odpovídá frekvenční charakteristika z měření č.9. Ani druhá frekvenční charakteristika se mnoho neodchyluje od frekvenční charakteristiky simulované soustavy. Vyznačené body o frekvenci identifikace se pro simulovanou soustavu i SOTD modely shodují. Body o frekvenci ω_2 se vzájemně mírně liší a bod z 9. měření byl určen lépe.



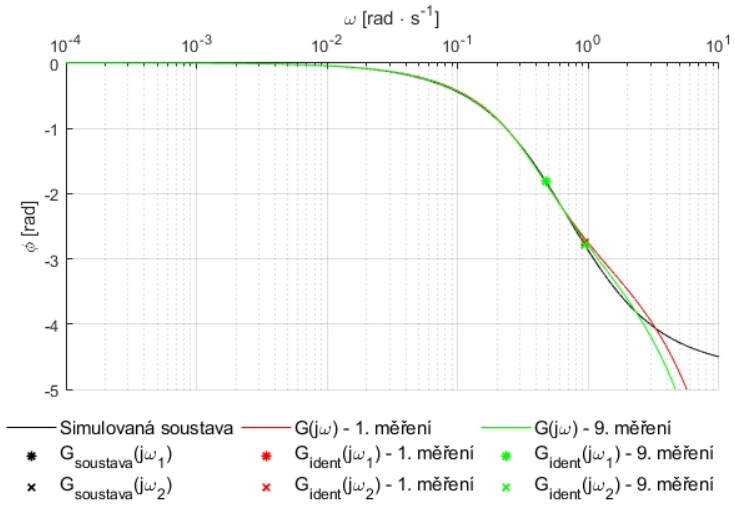
Obrázek 11.1: Nyquistovy frekvenční charakteristiky pro dva nejodlišnější výsledky měření, $\omega_1 = 0,469595\ rad/s$ a $\omega_2 = 0,93919\ rad/s$.

Amplitudové frekvenční charakteristiky obou modelů se blíží amplitudové charakteristice simulované soustavy (obr. 11.2). Amplitudové charakteristiky modelů jsou téměř totožné, avšak body vyznačené na frekvenční charakteristice z 1. měření se s body frekvenční charakteristiky simulované soustavy nepřekrývají tak dobře, jako je tomu u charakteristiky z 9. měření.

Na fázové frekvenční charakteristice je mezi těmito dvěma měřeními patrný rozdíl u frekvencí vyšších než $0,7\ rad/s$ (obr. 11.2). Frekvenční charakteristika modelu z 9. měření opět kopíruje frekvenční charakteristiku simulované soustavy lépe než model z 1. měření.



Obrázek 11.2: Amplitudové frekvenční charakteristiky pro dva nejodlišnější výsledky měření, $\omega_1 = 0,469595\ rad/s$ a $\omega_2 = 0,93919\ rad/s$.



Obrázek 11.3: Fázové frekvenční charakteristiky pro dva nejodlišnější výsledky měření, $\omega_1 = 0,469595\ rad/s$ a $\omega_2 = 0,93919\ rad/s$.

Pro oba modely, které se vzájemně příliš neliší, je možné říci, že soustava byla identifikována správně.

11.2 Testování na reálných soustavách

11.2.1 Identifikace soustavy „Teplovzdušný model“ – zapojení s ventilátorem

Program pro reléovou identifikaci byl použit k určení modelu laboratoní úlohy „Teplovzdušný model“ – zapojení s ventilátorem. Identifikace probíhala v pracovním bodě daném hodnotou akční veličiny $u_0 = 2,3\ V$Body určené identifikací jsou v tabulce 11.3, parametry modelu pak v tabulce 11.4.

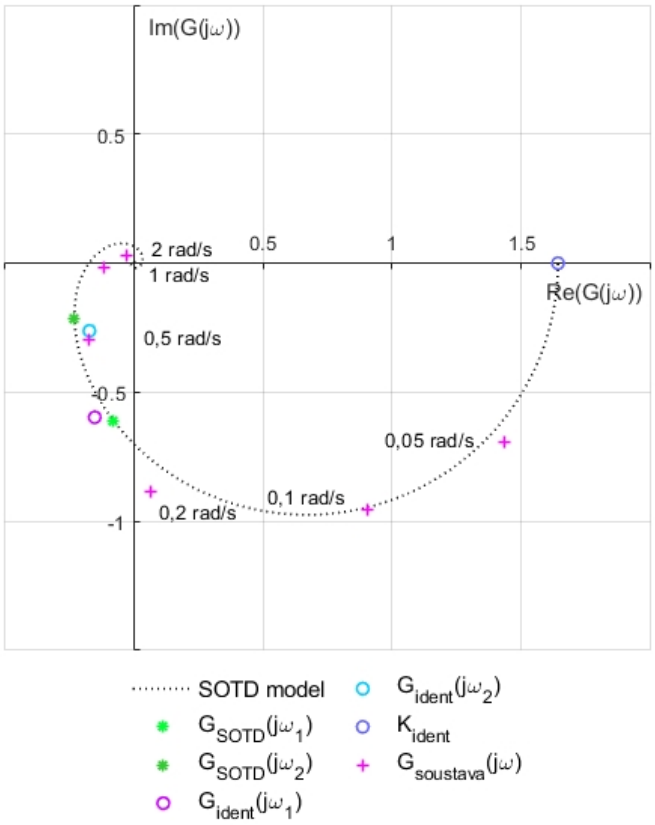
K	$\omega_1\ [rad/s]$	$G(j\omega_1)$	$\omega_2\ [rad/s]$	$G(j\omega_2)$
1,64209	0.2057	-0,151014-0,597077i	0,4114	-0,171728-0,262299i

Tabulka 11.3: Reléovou identifikací určené body frekvenčních charakteristik.

K	a_1	a_1	T_d
1,64209	12,6377	9,64180	1,70382

Tabulka 11.4: Určené parametry SOTD modelu.

Pro ověření správnosti modelu byly naměřeny body frekvenčních charakteristik soustavy a frekvenční charakteristiky modelu byly s těmto body porovnány.



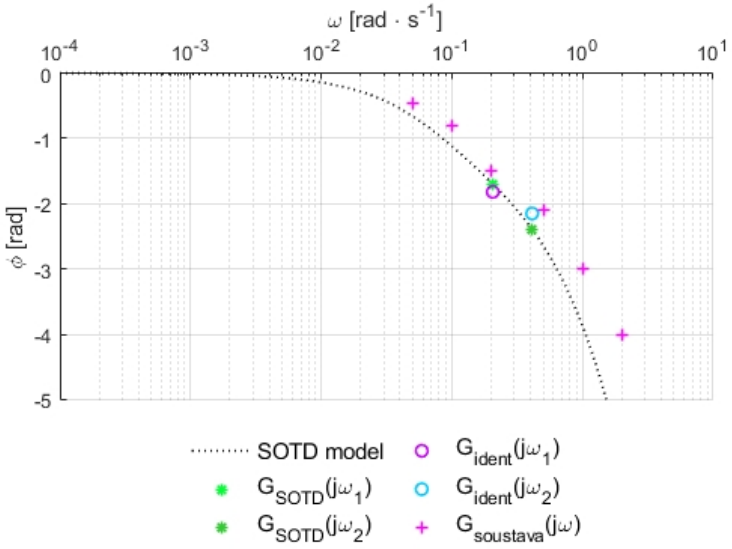
Obrázek 11.4: Porovnání Nyqustivy frekvenční charakteristiky určeného modelu s body frekvenční charakteristiky soustavy.

Nyquistova frekvenční charakteristika SOTD modelu (obr. 11.4) téměř kopíruje naměřené body frekvenční charakteristiky soustavy.

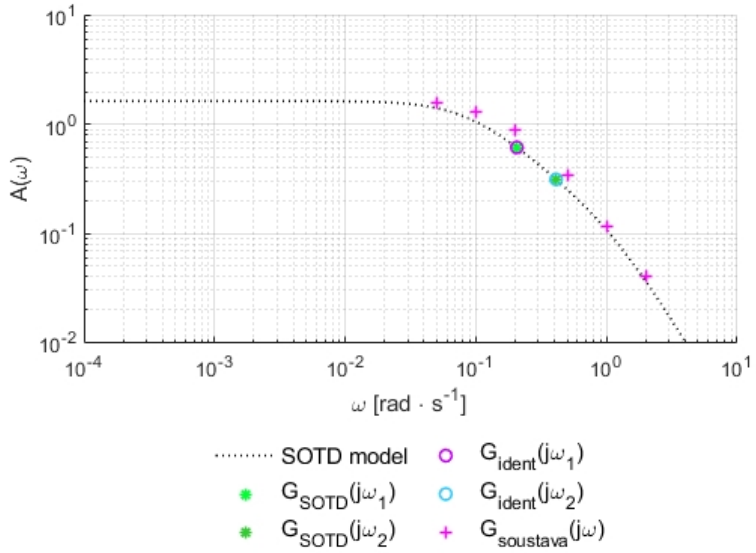
Fázová frekvenční charakteristika (obr. 11.5) ukazuje, že fáze začíná klesat o něco dříve než fáze soustavy. Nejedná se však o žádnou výraznou odchylku.

Amplitudová frekvenční charakteristika (obr. 11.6) má stejný průběh jako je ten, který naznačují body frekvenční charakteristiky soustavy.

Z frekvenčních charakteristik je možné soudit, že SOTD model byl určen s dostatečnou přesností pro frekvence nižší než $0,4\ rad/s$. Pak se již lehce odchyluje od chování reálné soustavy.



Obrázek 11.5: Porovnání fázové frekvenční charakteristiky určeného modelu s body frekvenční charakteristiky soustavy.



Obrázek 11.6: Porovnání amlitudové frekvenční charakteristiky určeného modelu s body frekvenční charakteristiky soustavy.

11.2.2 Identifikace soustavy „Teplovzdušný model“ – zapojení se žárovkou

Také pro soustavu „Teplovzdušný model“ – zapojení se žárovkou byl programem pro reléovou identifikaci určen SOTD model. Identifikace probíhala v pracovním bodě daném hodnotou akční veličiny $u_0 = 3\ V$. Body určené identifikací jsou v tabulce

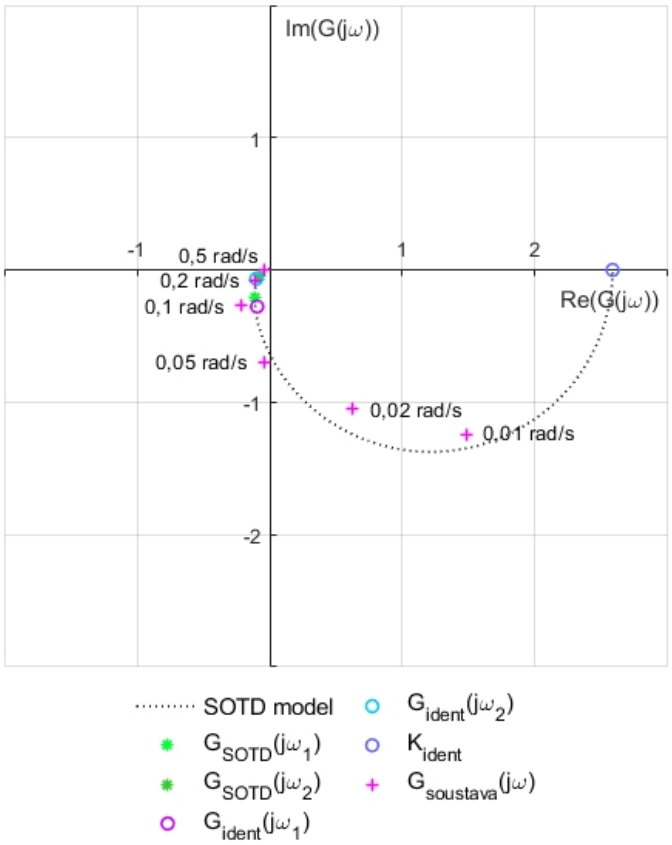
11.5, parametry modelu pak v tabulce 11.6.

K	ω_1 $[rad/s]$	$G(j\omega_1)$	ω_2 $[rad/s]$	$G(j\omega_2)$
2,586918	0,1277	-0,0964430-0,276906i	0,2554	-0,103205-0,0667331i

Tabulka 11.5: Reléovou identifikací určené body frekvenční charakteristiky.

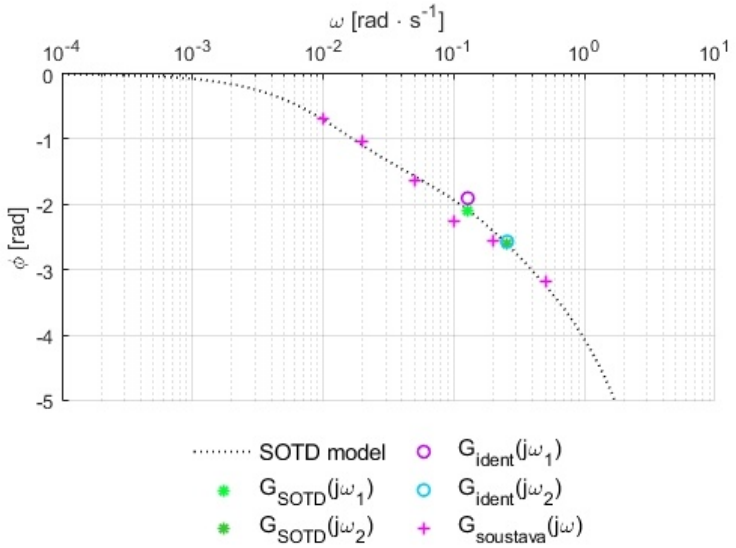
K	a_1	a_1	T_d
2,586918	80,8290	302,730	1,18257

Tabulka 11.6: Určené parametry SOTD modelu.



Obrázek 11.7: Porovnání Nyquistovy frekvenční charakteristiky určeného modelu s body frekvenční charakteristiky soustavy.

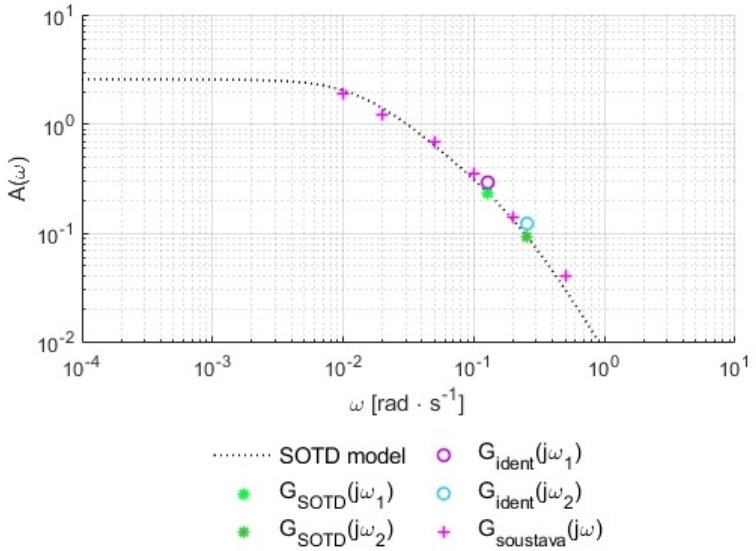
Porovnání Nyquistových frekvenčních charakteristik SOTD modelu a soustavy je na obrázku 11.7. Frekvenční charakteristika modelu svým tvarem dobře kopíruje polohu bodů Nyquistovy frekvenční charakteristiky soustavy.



Obrázek 11.8: Porovnání fázové frekvenční charakteristiky určeného modelu s body frekvenční charakteristiky soustavy.

Fázová frekvenční charakteristika modelu má velmi podobný průběh tomu, který je dán body frekvenční charakteristiky soustavy (obr. 11.8).

Amplitudová frekvenční charakteristika modelu odpovídá bodům amplitudové frekvenční charakteristiky soustavy (obr. 11.9). Z porovnání frekvenčních charakteristik SOTD modelu a bodů frekvenčních charakteristik soustavy lze tedy říci, že soustava byla dobře identifikována.



Obrázek 11.9: Porovnání amplitudové frekvenční charakteristiky určeného modelu s body frekvenční charakteristiky soustavy.

Kapitola 12

Určení modelu pro nastavení parametrů PI, PD a PID regulátorů

12.1 Testování na simulované soustavě

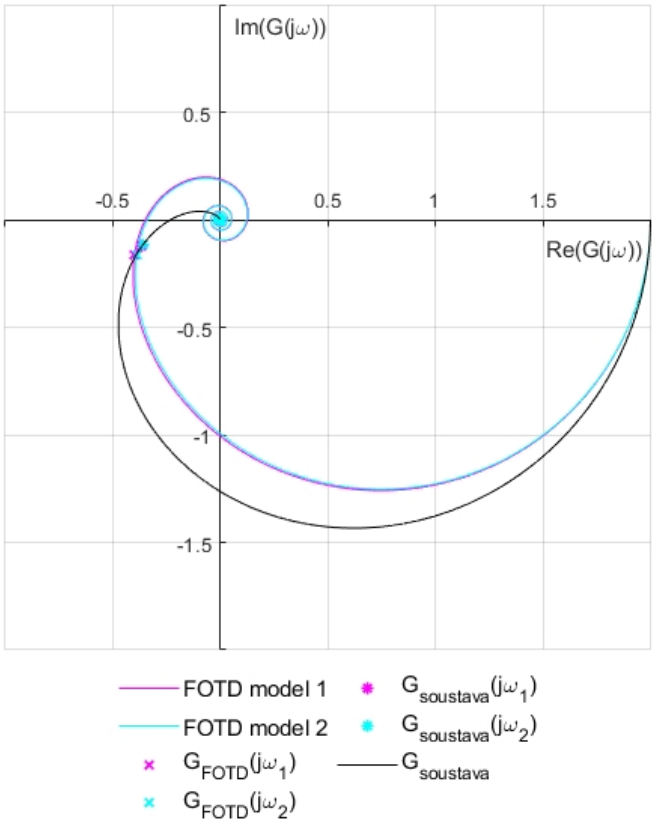
12.1.1 Opakovatelnost měření

Pro ověření správné funkce programu pro reléovou identifikaci bylo desetkrát opakováno měření na simulované soustavě za stejných podmínek. Velikost akční veličiny v pracovním bodě byla $u_0 = 3$. Z měření vyšly dva různé výsledky v závislosti na určené periodě kmitu (tab. 12.1). Stejně jako u předchozích dvou metod je to způsobené použitím funkce *GetRTC*.

	T_p	$u_{HighValue}$	$u_{LowValue}$	T	T_d	K
1	6,540	10	0	4,72924	1,45050	2
2	6,530	10	0	4,82535	1,43574	2
3	6,540	10	0	4,72924	1,45050	2
5	6,540	10	0	4,72924	1,45050	2
6	6,530	10	0	4,82535	1,43574	2
7	6,540	10	0	4,72924	1,45050	2
8	6,530	10	0	4,82535	1,43574	2
9	6,530	10	0	4,82535	1,43574	2
10	6,540	10	0	4,72924	1,45050	2

Tabulka 12.1: Určené parametry modelu pro 10 různých měření a parametry identifikace. T_p je perioda kmitu v s, $u_{HighValue}$ je ve V, $u_{LowValue}$ je ve V, T je v s a T_d je v s.

Pro porovnání rozdílu mezi oběma variantami řešení byly vykresleny frekvenční charakteristiky.

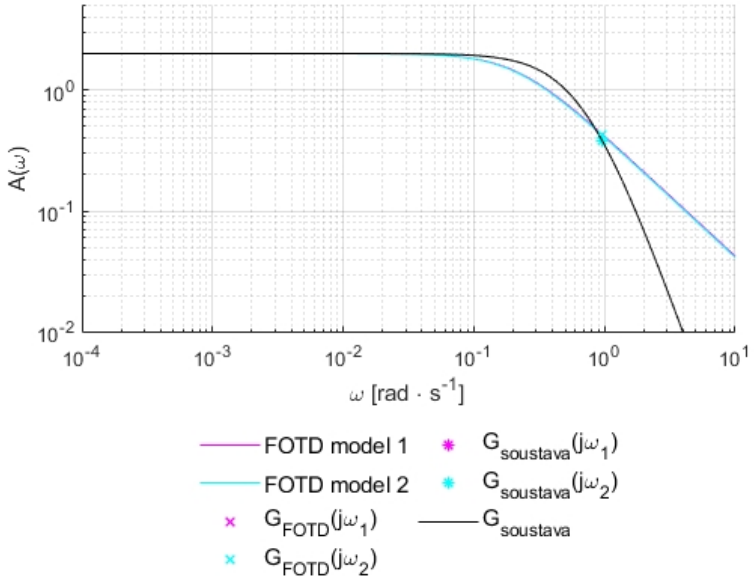


Obrázek 12.1: Porovnání Nyquistovy frekvenční charakteristiky simulované soustavy a frekvenčních charakteristik určených FOTD modelů.

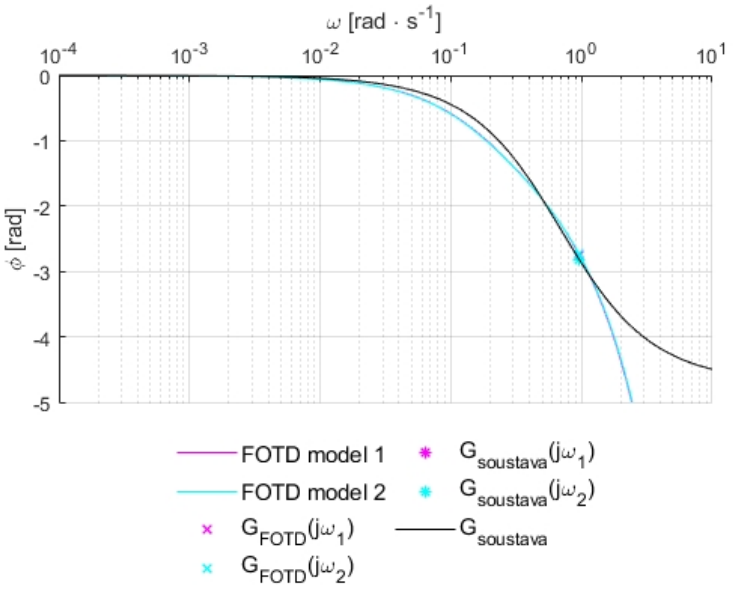
Nyquistovy frekvenční charakteristiky modelu 1 a modelu 2 jsou téměř stejné (12.1). I vzdálenost bodu modelu s frekvencí identifikace od příslušního bodu charakteristiky soustavy je srovnatelná. Frekvenční charakteristika soustavy je však odlišná od Nyquistových frekvenčních charakteristik modelů. Model popisuje frekvenční chování soustavy správně pouze při frekvencích blízkých frekvenci identifikace.

Amplitudové frekvenční charakteristiky modelů se téměř dokonale překrývají, avšak amplitudu soustavy určují správně jen při frekvencích blízkých frekvenci identifikace (obr. 12.2). Vzhledem k tomu, že model je prvního řádu a identifikovaná soustava třetího, jde o očekávaný výsledek.

Fázové frekvenční charakteristiky obou modelů se téměř dokonale překrývají (obr. 12.3). Také poměrně dobře sledují frekvenční charakteristiku soustavy až do frekvence, s jakou probíhala identifikace.



Obrázek 12.2: Porovnání amplitudové frekvenční charakteristiky simulované soustavy a frekvenčních charakteristik určených FOTD modelů.



Obrázek 12.3: Porovnání fázové frekvenční charakteristiky simulované soustavy a frekvenčních charakteristik určených FOTD modelů.

12.1.2 Správnost identifikace simulované soustavy

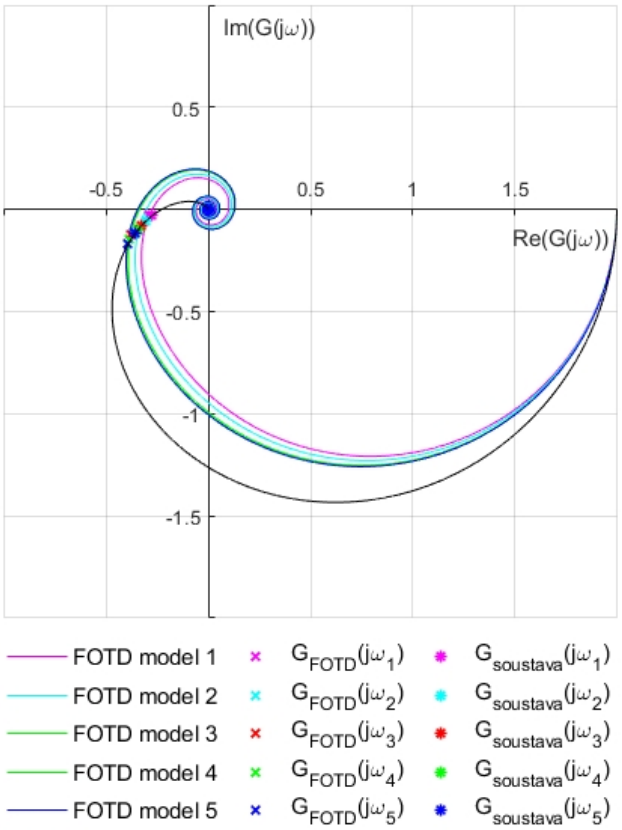
Protože přesnost identifikace může záviset na velikosti šumu – tedy na velikosti hystereze, byla provedena měření na simulované soustavě pro pět různých hodnot

velikosti šumu. V tabulce 12.2 lze vidět, že určené parametry FOTD modelu se liší jen mírně.

šum	T_p	$u_{HighValue}$	$u_{LowValue}$	T	T_d	K
0,01	5,590	7,29449	0	6,05548	1,43719	2
0,01	5,570	7,29449	0	5,83223	1,13752	2
0,02	5,940	9,97783	0	5,22254	1,38439	2
0,03	6,150	10	0	4,79856	1,42614	2
0,04	6,350	10	0	4,72848	1,44004	2
0,05	6,540	10	0	4,72924	1,45050	2

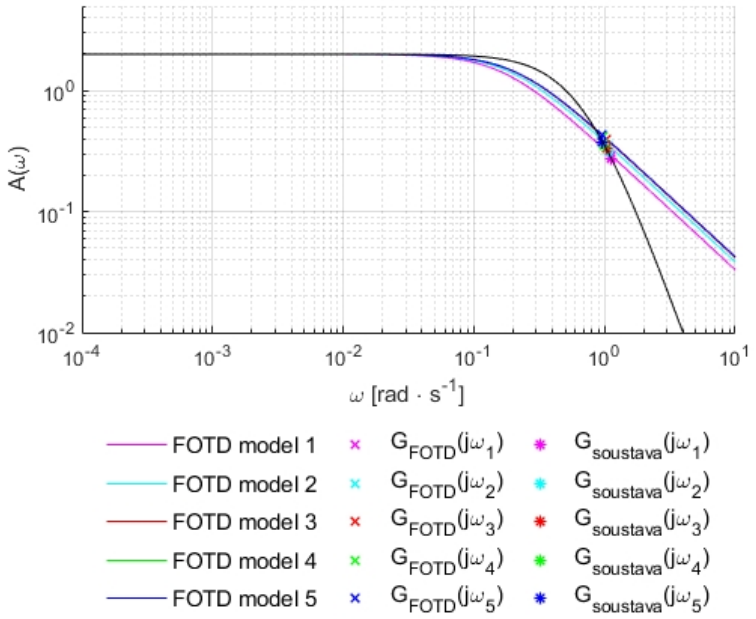
Tabulka 12.2: Porovnání parametrů modelu pro 5 různých hodnot šumu.

Pro lepší porovnání byly vykresleny frekvenční charakteristiky. Na nich jsou vyznačeny body s frekvencemi, které odpovídají frekvenci identifikace.

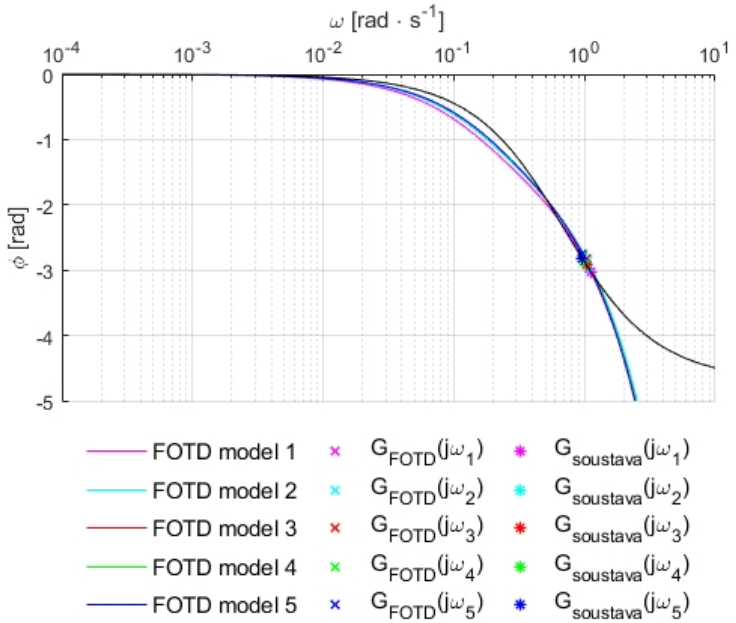


Obrázek 12.4: Porovnání Nyquistovy frekvenční charakteristiky simulované soustavy (černě) a frekvenčních charakteristik určených FOTD modelů.

Nyquistovy charakteristiky určených modelů si jsou velmi podobné, avšak od charakteristiky simulované soustavy se odlišují.



Obrázek 12.5: Porovnání amplitudové frekvenční charakteristiky simulované soustavy (černě) a frekvenčních charakteristik určených FOTD modelů.



Obrázek 12.6: Porovnání fázové frekvenční charakteristiky simulované soustavy (černě) a frekvenčních charakteristik určených FOTD modelů.

Amplitudové frekvenční charakteristiky určených modelů jsou téměř totožné. Mají však jiný sklon než frekvenční charakteristika simulované soustavy. Frekvenční

charakteristiky se protínají v bodě o frekvenci blízké frekvenci identifikace. V bodě identifikace je tedy amplituda určena dostatečně přesně.

Fázová frekvenční charakteristika simulované soustavy má nejprve mírný sklon. Sklon frekvenčních charakteristik určených modelů je výraznější. Později křivky klesají s podobným sklonem.

12.2 Testování na reálných soustavách

12.2.1 Identifikace soustavy „Teplovzdušný model“ – zapojení s ventilátorem

K soustavě „Teplovzdušný model“ bylo připojeno PLC v konfiguraci pro měření s ventilátorem. Velikost akční veličiny v pracovním bodě $u_0 = 2,3\ V$. Parametry FOTD modelu určené programem pro reléovou identifikaci jsou uvedené v tabulce 12.3.

K	T	T_d	T_p	$\omega\ [rad/s]$
1,75561	6,741670	3,00000	9,890	0,635307

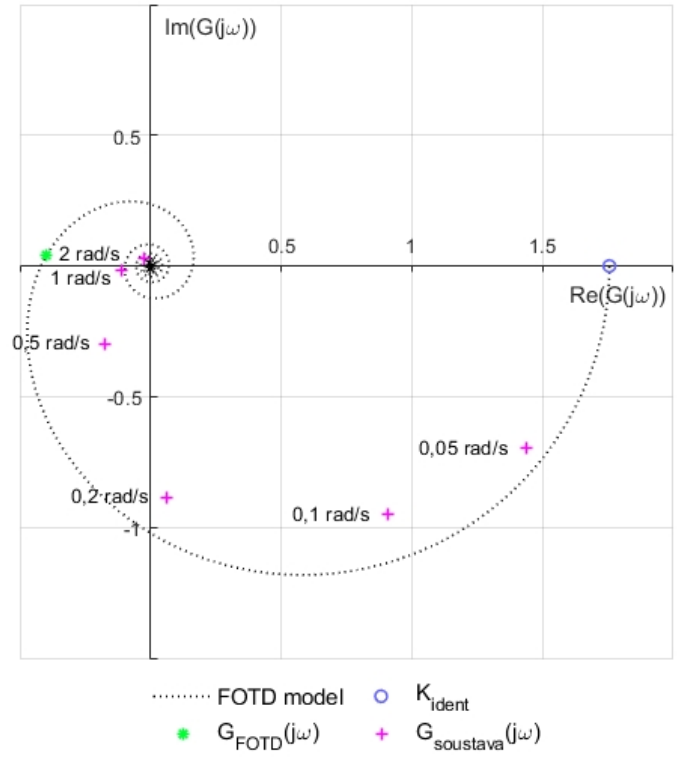
Tabulka 12.3: Parametry určeného FOTD modelu, perioda a frekvence identifikace.

Frekvenční charakteristiky určeného modelu byly porovnány s body frekvenční charakteristiky soustavy. Na nich jsou zobrazeny body s frekvencí odpovídající frekvenci identifikace.

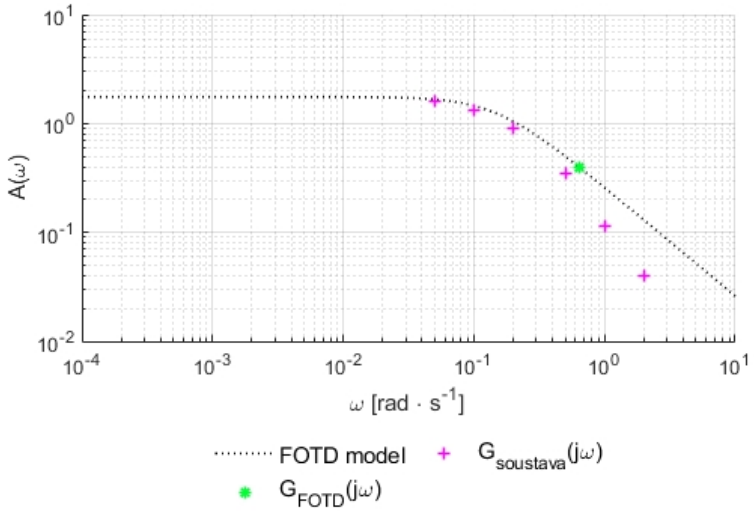
Nyquistova frekvenční charakteristika nalezeného modelu (obr. 12.7) je blízká frekvenční charakteristice soustavy pouze ve čtvrtém kvadrantu. Frekvenční chování modelu, jak je zobrazeno v Nyquistově frekvenční charakteristice, odpovídá soustavě s výraznějším dopravním zpožděním, než reálná soustava skutečně má. Jde však o jev, který se vyskytl i při identifikaci simulované soustavy. Zde je však patrnější.

Amplitudová frekvenční charakteristika modelu (obr. 12.8) má průběh, který napovídá, že statická citlivost soustavy byla určena správně. Sklon frekvenční charakteristiky modelu je pozvolnější, než na jaký je možné usuzovat z bodů frekvenční charakteristiky reálné soustavy.

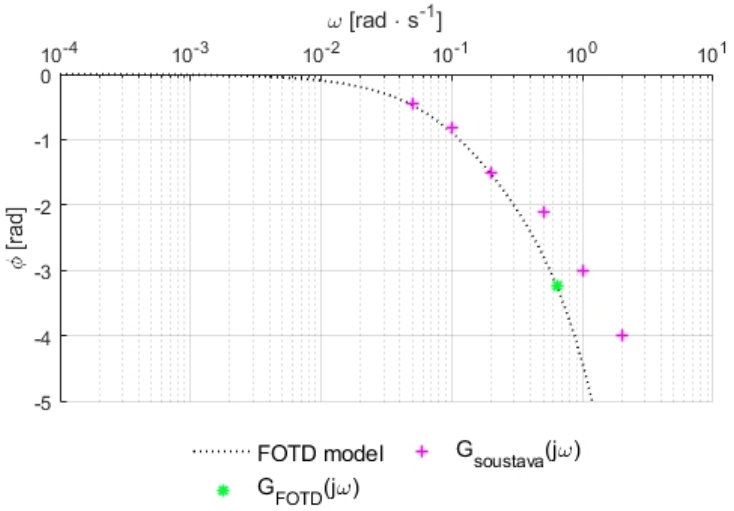
Fázová frekvenční charakteristika reálné soustavy (obr. 12.9) má mírnější sklon než frekvenční charakteristika určeného modelu.



Obrázek 12.7: Porovnání Nyquistovy frekvenční charakteristiky modelu s body Nyquistovy frekvenční charakteristiky soustavy.



Obrázek 12.8: Porovnání amplitudové frekvenční charakteristiky modelu s body amplitudové frekvenční charakteristiky soustavy.



Obrázek 12.9: Porovnání fázové frekvenční charakteristiky modelu s body fázové frekvenční charakteristiky soustavy.

12.2.2 Identifikace soustavy „Teplovzdušný model“ – zapojení se žárovkou

Soustava „Teplovzdušný tunel“ v zapojení se žárovkou byla identifikována v pracovním bodě daném hodnotou akční veličiny $u_0 = 3\ V$. Parametry určeného modelu jsou v tabulce 12.4.

K	T	T_d	T_p	$\omega\ [rad/s]$
2,68891	7,67907	4,97777	16,480	0,381261

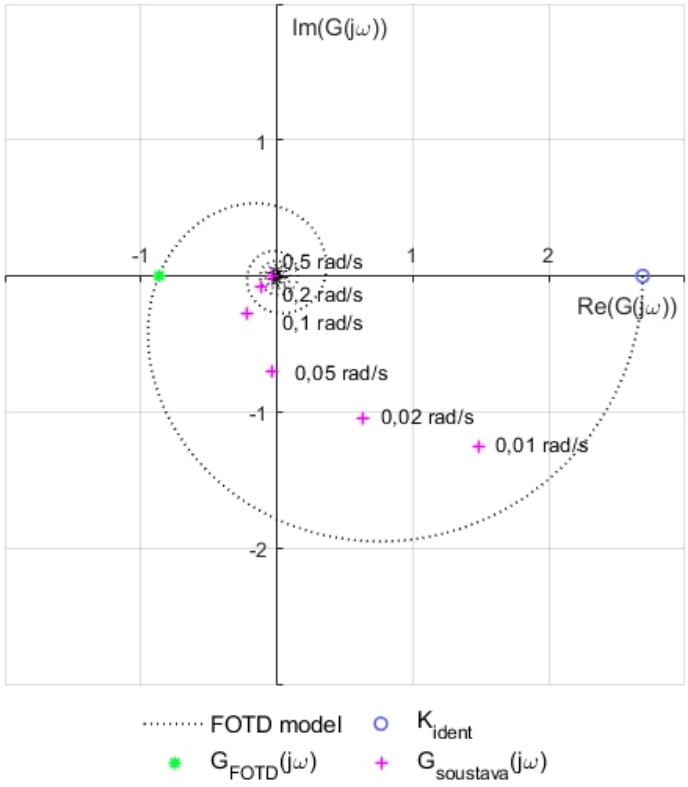
Tabulka 12.4: Parametry určeného FOTD modelu, perioda a frekvence identifikace.

Pro určený model byly vykresleny frekvenční charakteristiky pro porovnání s body frekvenční charakteristiky soustavy. Na frekvenčních charakteristikách jsou vyznačeny body odpovídající svou frekvencí frekvenci identifikace.

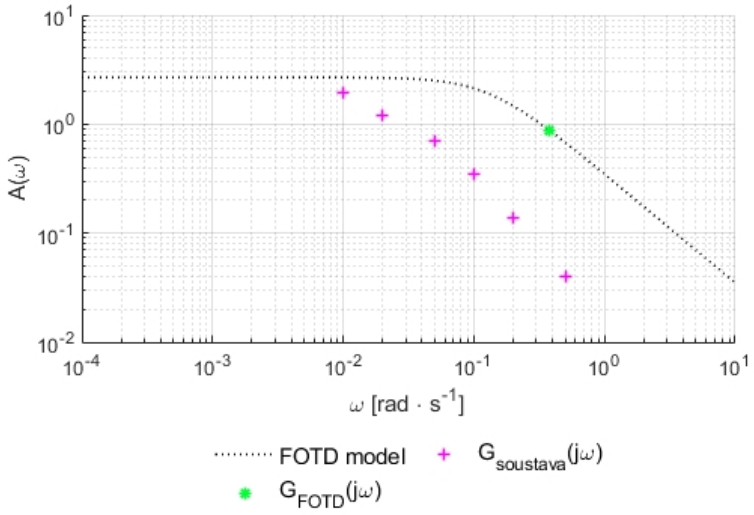
Průběh Nyquistovy frekvenční charakteristiky modelu ukazuje na větší dopravní zpoždění, než soustava skutečně vykazuje (obr. 12.10).

Amplitudová frekvenční charakteristika modelu má podobný sklon jaký pro soustavu naznačují body její amplitudové charakteristiky, avšak liší se hodnota časové konstanty, která je u soustavy zřejmě vyšší (obr. 12.11).

Fázová frekvenční charakteristika modelu je zcela jiná než by podle bodů frekvenční charakteristiky soustavy měla být, ale v okolí frekvence identifikace lze předpokládat, že se tyto frekvenční charakteristiky protínají (obr. 12.12).

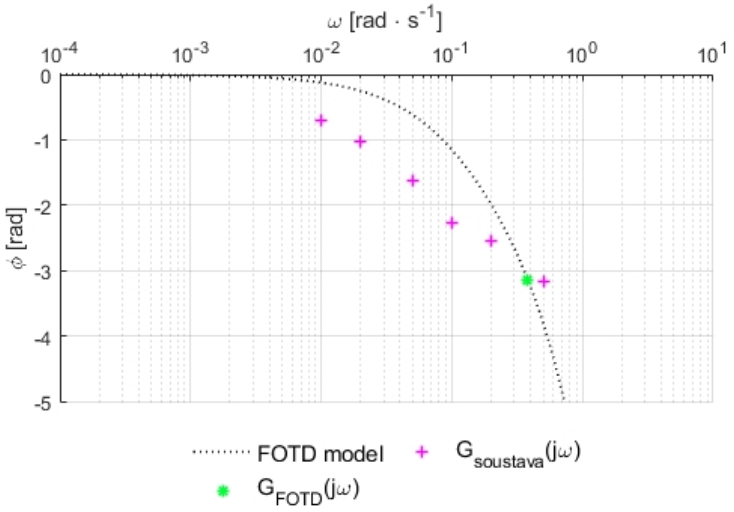


Obrázek 12.10: Porovnání frekvenční charakteristiky modelu s body Nyquistovy frekvenční charakteristiky soustavy



.

Obrázek 12.11: Porovnání frekvenční charakteristiky modelu s body amplitudové frekvenční charakteristiky soustavy



.

Obrázek 12.12: Porovnání frekvenční charakteristiky modelu s body fázové frekvenční charakteristiky soustavy

Kapitola 13

Porovnání použitých metod

Z použitých metod se ukázaly prakticky použitelnými dvě – Určení parametrů modelu ze tří bodů Nyquistovy frekvenční charakteristiky s přidaným dopravním zpožděním a Určení modelu pro nastavení parametrů PI, PD a PID regulátorů. Proto budou porovnány výsledky těchto dvou metod.

13.1 Kritérium správné polohy bodů Nyquistovy frekvenční charakteristiky

Pro porovnání správnosti identifikace bylo zvoleno kritérium správné polohy bodů Nyquistovy frekvenční charakteristiky ve tvaru

$$x_c(G_{model}, G_{soustava}) = \frac{\sum_{i=1}^{n} x_i(G_{model}, G_{soustava})}{n}, \tag{13.1}$$

kde n je počet frekvencí, pro které kontrolujeme polohu bodů frekvenční charakteristiky, a x_i je

$$x_i(G_{model}, G_{soustava}) = \frac{|G_{model}(j\omega_i) - G_{soustava}(j\omega_i)|}{G_{soustava}(j\omega_i)}. \tag{13.2}$$

13.1.1 Simulovaná soustava

Relativní odchylky polohy bodů Nyquistovy frekvenční charakteristiky modelu od charakteristiky simulované soustavy jsou uvedené v tabulce 13.1. Průměrné relativní odchylky pak jsou v tabulce 13.2. Je zřejmé, že SOTD model kopíruje Nyquistovu frekvenční charakteristiku simulované soustavy lépe než FOTD model. V žádném z vypočtených případů není odchylka bodu SOTD modelu větší než odchylka FOTD modelu. S rostoucí frekvencí přesnost SOTD modelu klesá. Vývoj přesnosti FOTD modelu není monotónní, avšak u nejvyšší frekvence dosahuje v celkovém porovnání nejvyšší odchylky. Průměrná relativní odchylka bodů FOTD modelu $0,5438$ je téměř šestkrát větší než relativní odchylka bodů SOTD modelu $0,0930$.

ω $[rad/s]$	$x_i(G_{SOTD}, G_{soustava})$	$x_i(G_{FOTD}, G_{soustava})$
0,05	0,0060	0,0815
0,1	0,0117	0,1501
0,2	0,0216	0,2330
0,5	0,0256	0,2302
1	0,0747	0,1918
2	0,4184	2,3762

Tabulka 13.1: Relativní odchylky bodů Nyquistovy frekvenční charakteristiky modelů od bodů frekvenční charakteristiky simulované soustavy.

$x_c(G_{SOTD}, G_{soustava})$	$x_c(G_{FOTD}, G_{soustava})$
0,0930	0,5438

Tabulka 13.2: Celkové relativní odchylky bodů Nyquistovy frekvenční charakteristiky modelů od bodů frekvenční charakteristiky simulované soustavy.

13.1.2 Soustava „Teplovzdušný model“ – zapojení s ventilátorem

Pro soustavu „Teplovzdušný model“ – zapojení s ventilátorem byly určeny relativní odchylky bodů Nyquistovy frekvenční charakteristiky modelů od bodů soustavy (13.3) a jejich průměrná hodnota (13.4).

ω $[rad/s]$	$x_i(G_{SOTD}, G_{soustava})$	$x_i(G_{FOTD}, G_{soustava})$
0,05	0,2279	0,0761
0,1	0,3396	0,0296
0,2	0,3251	0,0932
0,5	0,5302	0,7288
1	0,8206	1,9724
2	1,5827	4,0584

Tabulka 13.3: Relativní odchylky bodů Nyquistovy frekvenční charakteristiky modelů od bodů frekvenční charakteristiky soustavy „Teplovzdušný model“ – zapojení s ventilátorem.

Při frekvencích menších než $0,5\ rad/s$ vykazuje menší odchylky FOTD model. V ostatních frekvencích je přesnější SOTD model. Průměrná odchylka bodů FOTD modelu $1,1597$ je téměř dvakrát větší než průměrná relativní odchylka bodů SOTD modelu $0,6377$. Oproti simulované soustavě, kde jsou body Nyquistovy frekvenční charakteristiky soustavy určeny přesně, u reálné soustavy může být v důsledku chyby měření jejich poloha posunutá oproti skutečné. Uvažujme však, že tyto odchylky nejsou příliš velké.

$x_c(G_{SOTD}, G_{soustava})$	$x_c(G_{FOTD}, G_{soustava})$
0,6377	1,1597

Tabulka 13.4: Celkové relativní odchylky bodů Nyquistovy frekvenční charakteristiky modelů od bodů frekvenční charakteristiky soustavy „Teplovzdušný model“ – zapojení s ventilátorem.

13.1.3 Soustava „Teplovzdušný model“ – zapojení se žárovkou

Také pro identifikace soustavy „Teplovzdušný model“ v zapojení se žárovkou byly určeny relativní odchylky polohy bodů Nyquistovy frekvenční charakteristiky určeným modelů od bodů frekvenční charakteristiky soustavy.

ω $[rad/s]$	$x_i(G_{SOTD}, G_{soustava})$	$x_i(G_{FOTD}, G_{soustava})$
0,01	0,0567	0,7670
0,02	0,1678	1,6265
0,05	0,1045	3,1737
0,1	0,3252	5,7167
0,2	0,1482	9,6478
0,5	0,2454	16,1486

Tabulka 13.5: Relativní odchylky bodů Nyquistovy frekvenční charakteristiky modelů od bodů frekvenční charakteristiky soustavy „Teplovzdušný tunel“ – zapojení s ventilátorem

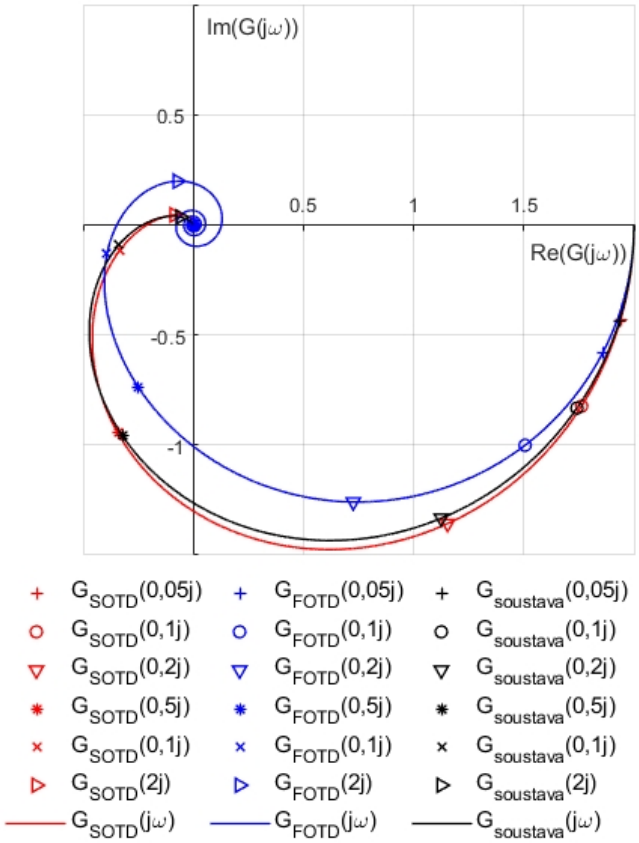
Vzhledem k nepřesně určenému FOTD modelu platí i v tomto případě, a zde je tento jev nejvýraznější, že relativní odchylky bodů frekvenční charakteristiky SOTD modelu jsou menší než odchylky FOTD modelu (tab. 13.5). V nejhorším případě je odchylka FOTD modelu téměř $95x$ větší než ochylka SOTD modelu. Logickým výstupem je, že průměrná relativní odchylka bodů SOTD modelu je více než $40x$ menší než průměrná relativní odchylka bodů FOTD modelu (tab. 13.6).

$x_c(G_{SOTD}, G_{soustava})$	$x_c(G_{FOTD}, G_{soustava})$
0,1746	6,1801

Tabulka 13.6: Celkové relativní odchylky bodů Nyquistovy frekvenční charakteristiky modelů od bodů frekvenční charakteristiky soustavy „Teplovzdušný tunel“ – zapojení se žárovkou

13.2 Porovnání frekvenčních charakteristik

13.2.1 Simulovaná soustava

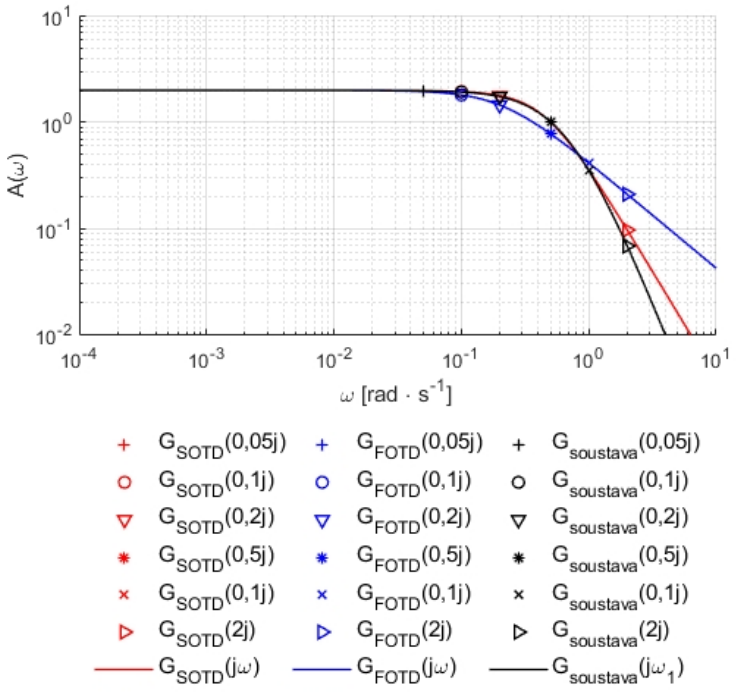


Obrázek 13.1: Porovnání Nyquistovy frekvenční charakteristiky simulované soustavy a frekvenčních charakteristik určených modelů.

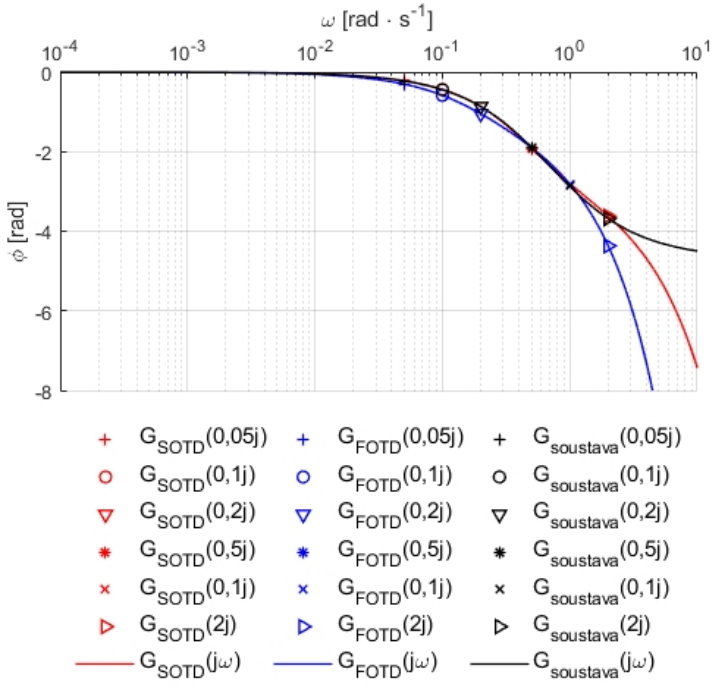
Na obrázku 13.1 jsou vykreslené Nyquistovy frekvenční charakteristiky modelů a soustavy. Na charakteristikách jsou vyznačené body, pro které byla v předchozí kapitole vyhodnocována přesnost polohy. SOTD model téměř odpovídá simulované soustavě. FOTD model je od soustavy odchýlený.

Amplitudová frekvenční charakteristika SOTD modelu odpovídá více charakteristice soustavy než FOTD model (obr. 13.2). Sklon amplitudových charakteristik je dán řádem soustavy. Amplitudová frekvenční charakteristika soustavy druhého řádu tedy logicky lépe aproximuje amplitudovou charakteristiku soustavy třetího řádu než soustava prvního řádu.

Fázová frekvenční charakteristika soustavy je dobře aproximována fázovými frekvenčními charakteristikami obou modelů asi do frekvence $\omega = 1\ rad/s$ (obr. 13.2). SOTD model pak kopíruje fázovou frekvenční charakteristiku soustavy až do frekvence $\omega = 2\ rad/s$.



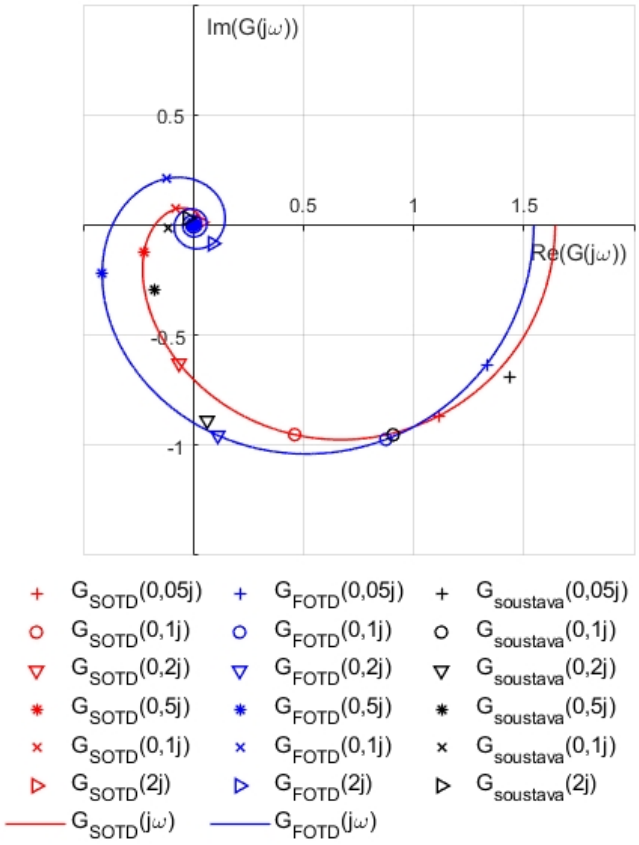
Obrázek 13.2: Porovnání amplitudové frekvenční charakteristiky simulované soustavy a frekvenčních charakteristik určených modelů.



Obrázek 13.3: Porovnání fázové frekvenční charakteristiky simulované soustavy a frekvenčních charakteristik určených modelů.

13.2.2 Soustava „Teplovzdušný model“ – zapojení s ventilátorem

Při porovnávání bodů frekvenčních charakteristik modelů a soustavy je třeba vzít v úvahu skutečnost, že body charakteristiky soustavy nemusí být určeny naprosto přesně. Navíc každá reálná soustava se může chovat odlišně v závislosti na stavu okolního prostředí, což je zde patrné na rozdílné hodnotě statické citlivosti (obr. 13.4). Ta by dle testů na simulované soustavě měla být určena přesně, avšak zde stejná metoda vedla ke dvěma rozdílným výsledkům. Rychlost růstu frekvence je u Nyquistovy frekvenční charakteristiky FOTD modelu bližší vývoji frekvence u charakteristiky soustavy než charakteristika SOTD modelu. Ta však má tvar bližší naměřeným bodům Nyquistovy frekvenční charakteristiky soustavy „Teplovzdušný model“ – zapojení s ventilátorem.

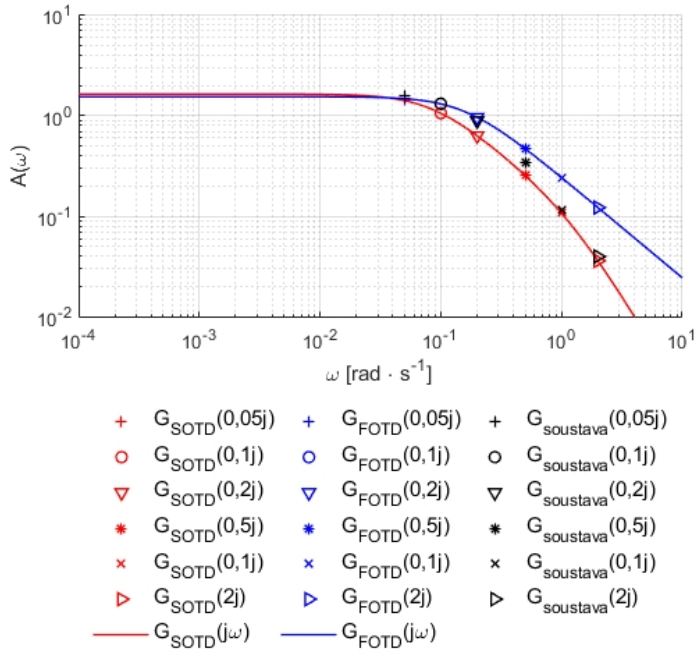


Obrázek 13.4: Porovnání Nyquistovy frekvenční charakteristiky soustavy „Teplovzdušný model“ – zapojení s ventilátorem a frekvenčních charakteristik určených modelů.

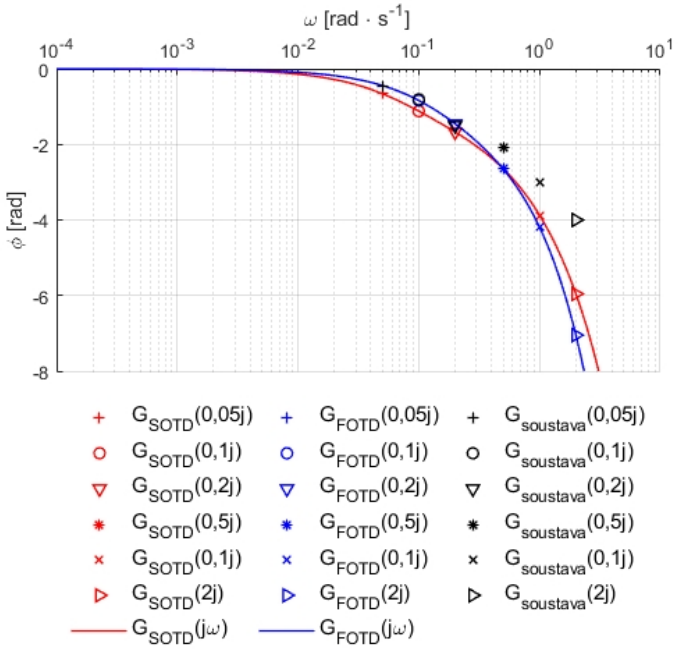
Amplitudová frekvenční charakteristika FOTD modelu kopíruje body frekvenční charakteristiky přibližně do frekvence $\omega = 0,4\ rad/s$ (obr. 13.5). Při růstu frekvence nad tuto hodnotu se od naměřených bodů frekvenční charakteristiky soustavy odchyluje a soustavu pak lépe aproximuje SOTD model.

Fázovou frekvenční charakteristiku soustavy kopíruje FOTD model do frekvence $\omega = 0,1\ rad/s$ a pak se se od ní odchyluje (obr. 13.6). Při vyšších frekvencích už je

bodům frekvenční charakteristiky soustavy bližší SOTD model.



Obrázek 13.5: Porovnání amplitudové frekvenční charakteristiky soustavy „Teplovzdušný model“ – zapojení s ventilátorem a frekvenčních charakteristik určených modelů.



Obrázek 13.6: Porovnání bodů fázové frekvenční charakteristiky soustavy „Teplovzdušný model“ – zapojení s ventilátorem a frekvenčních charakteristik určených modelů.

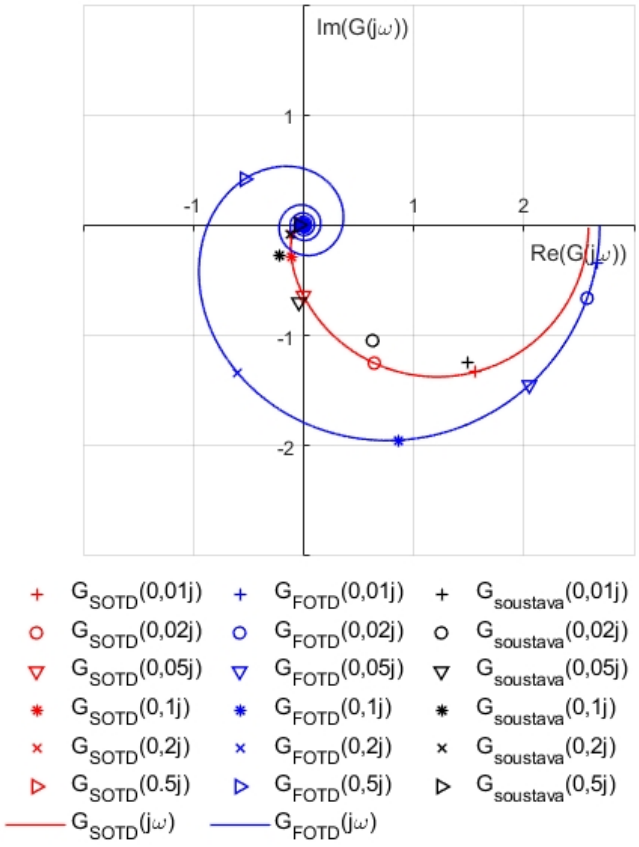
13.2.3 Soustava „Teplovzdušný model“ – zapojení se žárovkou

K porovnání přesnosti určených modelů byly vykresleny frekvenční charakteristiky obou modelů a vyznačeny body o stejných frekvencích, pro jaké byly naměřeny body frekvenčních charakteristik soustavy.

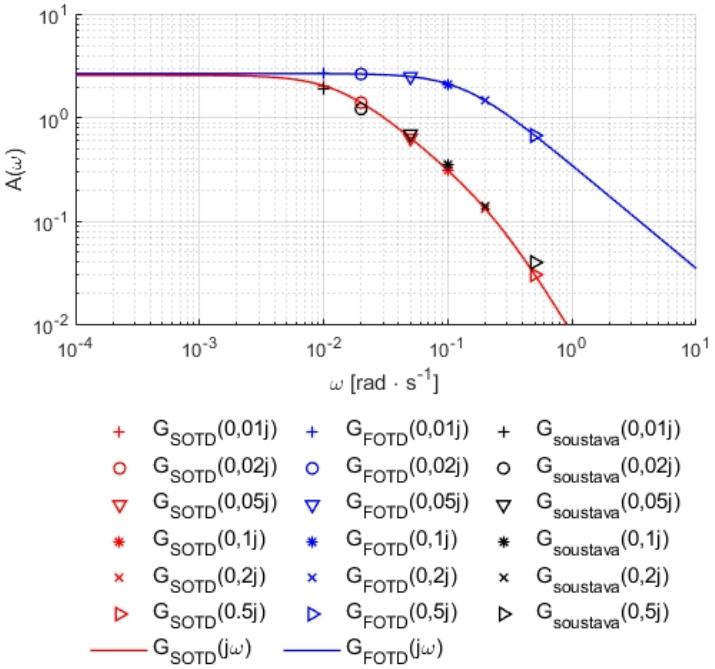
Nyquistova frekvenční charakteristika FOTD modelu se značně liší od bodů frekvenční charakteristiky soustavy (obr. 13.7). Zato určený SOTD model se velmi blíží naměřeným bodům frekvenční charakteristiky soustavy.

Z amplitudové frekvenční charakteristiky je patrné, že SOTD model popisuje závislost amplitudy kmitu soustavy na frekvenci (obr. 13.8). Z průběhů je možné usuzovat, že i časové konstanty SOTD modelu byly určeny dostatečně přesně. Oproti tomu je patrné, že časová konstanta FOTD modelu je nižší, než by podle bodů frekvenční charakteristiky soustavy měla být.

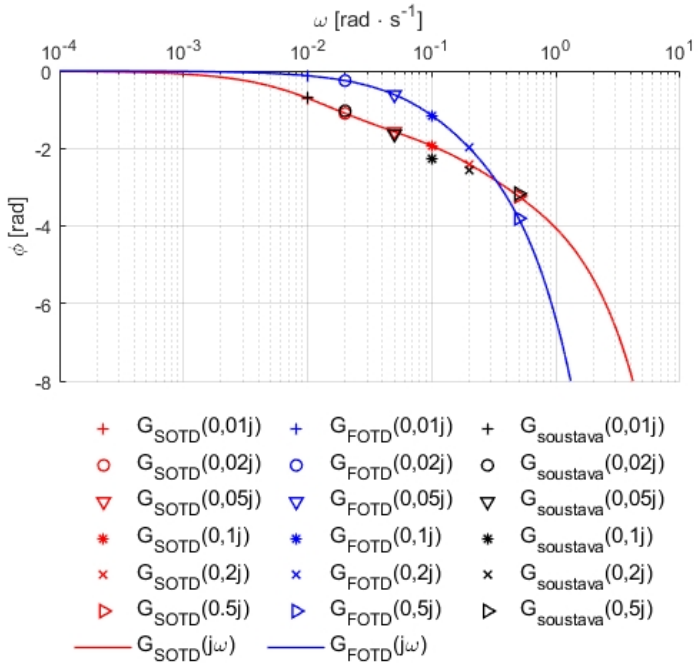
Fázová frekvenční charakteristika FOTD modelu popisuje soustavu dobře pouze v okolí frekvence $\omega = 0,3\ rad/s$ (obr. 13.9). SOTD model popisuje podle fázové frekvenční charakteristiky soustavu dobře.



Obrázek 13.7: Porovnání Nyquistovy frekvenční charakteristiky soustavy „Teplovzdušný model“ – zapojení se žárovkou a frekvenčních charakteristik určených modelů.



Obrázek 13.8: Porovnání amplitudové frekvenční charakteristiky soustavy „Teplovzdušný model“ – zapojení se žárovkou a frekvenčních charakteristik určených modelů.



Obrázek 13.9: Porovnání bodů fázové frekvenční charakteristiky soustavy „Teplovzdušný model“ – zapojení se žárovkou a frekvenčních charakteristik určených modelů.

Část V

Příprava funkčního bloku reléové identifikace do knihovny iControlLib

Funkční blok je jednou ze tří organizačních jednotek pro programování PLC. Dalšími organizačními jednotkami jsou funkce a programy.

13.3 Funkční blok dle standardu IEC EN 61 131–3

Pokud postupujeme podle standardu IEC EN 61 131–3, pak je funkčním blokem myšlena organizační jednotka programu, která po proběhnutí vrátí jednu nebo i více hodnot. Tím se odlišuje od funkce, která vrací pouze jednu hodnotu. Při použití funkčního bloku v programu se vytváří jeho instance. Tedy kopie, která má přiřazený identifikátor – jméno a také datovou strukturu. Ta obsahuje vstupní, vnitřní a výstupní proměnné, které mohou být i typu struktura. Všechny hodnoty proměnných funkčního bloku se v paměti zachovávají, dokud je nepřepíše další chod funkčního bloku. Instance funkčního bloku je třeba vytvořit deklarací ve třídě VAR nebo VAR_GLOBAL. Pokud funkční blok není deklarován jako globální, je dostupný pouze v té organizační jednotce (funkce, funkční blok, program), kde je deklarován. V rámci této jednotky může být používán opakovaně [11].

Při použití funkčního bloku vidíme hodnoty pouze vstupních a výstupních veličin. Hodnoty vnitřních veličin nejsou přístupné. Dále není povoleno přiřadit hodnotu výstupu funkčního bloku. To může provést jen funkční blok sám. Přiřazení hodnoty vstupu funkčního bloku je možné provést kdykoliv v nadřazené organizační jednotce a je typicky součástí volání funkčního bloku.

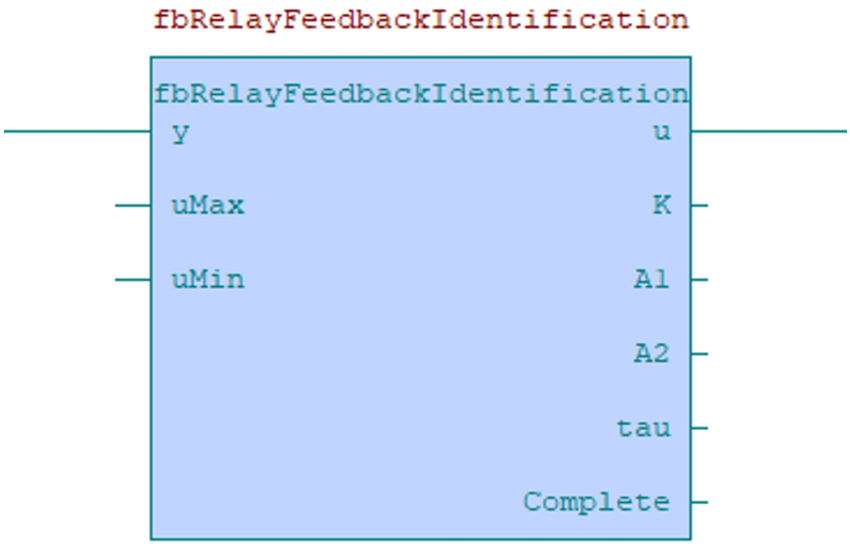
13.4 Deklarace funkčního bloku

Deklarace funkčního bloku začíná klíčovým slovem $FUNCTION\_BLOCK$ a končí klíčovým slovem $END\_FUNCTION\_BLOCK$. Funkční blok může mít více výstupních parametrů. Ty se deklarují ve třídě VAR_OUTPUT. Proměnné předávané funkčnímu bloku deklarované ve třídě VAR_INPUT není možné uvnitř funkčního bloku měnit. K tomu slouží proměnné deklarované jako VAR_IN_OUT nebo $VAR\_EXTERNAL$. Hodnoty takových proměnných mohou být uvnitř funkčního bloku měněny. K deklaraci vstupních proměnných je také možné použít kvantifikátory R_EDGE a F_EDGE. Ty označují funkci pro detekci hran a jsou určeny pro vstupy typu $BOOL$. Pro deklaraci hodnot vstupů funkčního bloku se používá stejná konstrukce jako pro inicializaci funkcí [11].

13.5 Funkční blok reléové identifikace

Z programu reléové identifikace se funkční blok vytvoří předeklarováním proměnných a změnou klíčových slov programu na klíčová slova funkčního bloku. Proměnné, které se zadávaly pomocí webového rozhranní, je třeba uložit jako prvky třídy VAR_INPUT stejně jako hodnotu regulované veličiny y. Výstupy, tedy prvky třídy VAR_OUTPUT, budou parametry modelu a akční veličina u. Dále je třeba mezi výstupy funkčního bloku přidat proměnnou typu $BOOL$ ukazující na ukončení identifikace. Takto vytvořený funkční blok je možné vložit do knihovny Mosaicu. Je

však třeba zajistit, aby měl dostupné funkční bloky a funkce, které sám používá. To je možné řešit přidáním funkčních bloků *fbstabilization*, *fbWebInputControl* a *fbvypocetparametru* do jiné knihovny, která bude vázána ke knihovně iControlLib, nebo přidáním funkčních bloků přímo do knihovny. Vytvořený funkční blok reléové identifikace ze tří bodů Nyquistovy frekvenční charakteristiky se jmenuje *fbRelayFeedbackIdentification* a po vložení do knihovny iControlLib [19] spolu s ostatními funkčními bloky, je připravený k používání v dalších programech.



Obrázek 13.10: Zobrazení vytvořeného funkčního bloku *fbRelayFeedbackIdentification* v jazyce FBD.

Část VI

Závěr

V rámci této práce byly vybrány tři metody reléové identifikace k naprogramování pro PLC Tecomat Foxtrot. Jedná se o určení parametrů modelu ze tří bodů Nyquistovy frekvenční charakteristiky [7], určení parametrů modelu ze tří bodů Nyquistovy frekvenční charakteristiky s přidaným dopravním zpožděním [8] a určení modelu pro nastavení parametrů PI, PD a PID regulátorů [9]. Zvolené metody jsou vhodné pro identifikaci SISO systémů. Pro komunikaci mezi PLC a soustavou tedy slouží jeden analogový vstup a jeden analogový výstup se shodným napěťovým rozsahem $0 - 10\ V$.

Podle zvolených metod byly napsány programy reléové identifikace v jazyce structured text, který podléhá standardu IEC EN 61 131–3. Jejich výstupem je u metody identifikace ze tří bodů frekvenční charakteristiky s dopravním zpožděním i bez dopravního zpoždění SOTD model. U identifikace k určení modelu pro výpočet parametrů PI, PD nebo PID regulátorů pak FOTD model. Jde o modely s dopravním zpožděním. Přidání dopravního zpoždění k modelu nižšího řádu při identifikaci soustavy vyššího řádu je běžným postupem.

Napsané programy byly testovány na simulované soustavě třetího řádu a na laboratorní soustavě Teplovzdušný model ve dvou variantách zapojení – v zapojení s ventilátorem a se žárovkou. Při použití určení parametrů modelu ze tří bodů frekvenční charakteristiky byly určeny body Nyquistovy frekvenční charakteristiky. Jejich poloha však byla taková, že výpočet parametrů modelu selhal a bylo by třeba hledat jinou cestu pro určení parametrů modelu. Nejspíše by bylo nutné použít některou numerickou metodu výpočtu. Tím bychom ale přišli o jednoduchost metody, která je jedním ze základních požadavků pro práci s PLC. Při použití vylepšené metody určení parametrů modelu ze tří bodů Nyquistovy frekvenční charakteristiky s přidaným dopravním zpožděním byl model určen ve všech případech. Také metoda určení modelu pro výpočet parametrů PI, PD a PID regulátorů identifikovala soustavu ve všech případech.

Pro určené modely byla vykreslena porovnání jejich frekvenčních charakteristik a charakteristiky soustavy, respektive frevenčních charakteristik modelů a bodů frekvenčních charakteristik soustavy. Přesnost identifikace soustav byla posouzena pomocí kritéria pro správnost polohy bodů Nyquistovy frekvenční charakteristiky. Z těchto porovnání vychází jako lepší metoda určení parametrů modelu ze tří bodů Nyquistovy frekvenční charakteristiky s přidaným dopravním zpožděním. Skutečnost, že určení modelu pro výpočet parametrů PI, PD a PID regulátorů nenachází tak přesné modely, může být dána použitím FOTD modelu, který není schopný sledovat průběh amplitudy soustavy vyššího řádu tak dobře, jako SOTD model. K použití složitějšího modelu nebylo přikročeno proto, že pak by bylo nutné k výpočtu jeho parametrů použít některou z numerických metod výpočtu. Fázové frekvenční charakteristiky určených FOTD modelů odpovídaly charakteristikám soustav srovnatelným způsobem s fázovými frekvenčními charakteristikami SOTD modelů. Proto je možné usuzovat, ale nutné otestovat, zda by tato metoda neidentifikovala výrazně úspěšněji soustavy s chováním blízkým systémům prvního řádu.

Program určení parametrů modelu ze tří bodů frekvenční charakteristiky byl přepracován ve funkční blok pro vložení do knihovny iControlLib programovacího prostředí Mosaic.

Přílohy

Příloha A

Literatura

[1] ZIEGLER, John G., NICHOLS, Nathaniel B. *Optimum settings for automatic controllers*. Trans. ASME. 1943, Volume 65, str. 433–444.

[2] TSYPKIN, Yakov Zalmanovich. *Releinye autmatischeskie sistemy*. Vydavatelství Nauka, Moskva, 1974.

[3] ÄSTRÖM, Karl J., HÄGGLUND, Tore. *Advanced PID Control*. ISA - The Instrumentation, Systems and Automation Society, 2006.

[4] BERNER, Josefin, HÄGGLUND, Tore, ÄSTRÖM Karl J. *Improved relay autotuning using normalized time delay*. In *American Control Conference (ACC)*. Boston, 2016, str. 1869–1875.

[5] HOFREITER, Milan. *Shifting Method for Relay Feedback Identification*. In *IFAC Conference on Manufacturing Modelling, Management and Control*, 28.–30. červen, Troyes, 2016.

[6] CHIDAMBARAM M., Sathe V. *Relay Autotuning for Identification and Control*. Cambridge University Press, Cambridge, 2014.

[7] HOFREITER, Milan. *Biased-Relay Feedback Identification for Time Delay Systems*. IFAC-PapersOnLine, Elsevier. 2017, Volume 50, Issue 1, str. 15185–15190.

[8] HOFREITER, Milan. *Alternative Identification Method using Biased Relay Feedback*, přijato na The 2018 IFAC Symposium on Information Control Problems in Manufacturing (INCOM 2018), Bergamo, 2018.

[9] BERNER, Josefin. *Automatic Tuning of PID Controllers based on Asymmetric Relay Feedback*. Lund, 2015. Disertační práce. Lund university, Lund Institute of Technology, Department of Automatic Control, 2015.

[10] *PLC Tecomat Foxtrot – základní moduly. CP-1005, CP-1015.* [online].[cit.8.6.2018]. Dostupné z: https://www.tecomat.cz/modules/DownloadManager/download.php?alias=foxtrot-cz-cp-1005.

[11] *Nápověda programu Mosaic*. Verze 2017.2. [cit.8.6.2018]

[12] *Automa – časopis pro automatizační techniku: Esperanto programátorů PLC: programování podle normy IEC/EN 61131-3*. ISSN 1210-9592.

[13] HORNYCHOVÁ, Alžběta. *Odhad parametru anisochronního modelu ze zadaných bodu frekvenční charakteristiky*. Praha. Bakalářská práce. ČVUT v Praze, Fakulta strojní. Vedoucí práce Milan Hofreiter, 2016.

[14] HORNYCHOVÁ, Alžběta. *Určení parametrů anisochronního modelu soustavy pomocí releové identifikace a diferenciální evoluce*. In *Studentská tvůrčí činnost*. Praha, 2017. ISBN 978-80-01-06421-4.

[15] HORNYCHOVÁ, Alžběta. *Reléová identifikace v uzavřené regulační smyčce pomocí programovatelného automatu Tecomat Foxtrot*. In *Studentská tvůrčí činnost*. Praha, 2018. ISBN 978-80-01-06421-4.

[16] *OSCAT BASIC:LIBRARY Documentation In English, Version 3.33*. [online].[cit.15.4.2018]. Dostupné z: https://www.tecomat.cz/uploads/files/sw/Mosaic/OSCAT/oscat_basic333_en.pdf.

[17] *Knihovna pro modelování procesů*. [online]. [cit.12.4.2018]. Dostupné z: https://www.tecomat.cz/modules/DownloadManager/download.php?alias=txv00344_01_mosaic_modellib_cz.

[18] *Teplovzdušný model. Návody na laboratorní cvičení z automatického řízení*. [online].[cit.19.3.2018]. Dostupné z: http://vlab.fs.cvut.cz/navody/files/tvm.pdf.

[19] *Knihovna iControlLib*. [online]. [cit.12.4.2018]. Dostupné z: https://www.tecomat.cz/uploads/files/DOCS/cze/TXV00359_01_Mosaic_iControlLib.pdf.

Příloha B

Seznam použitých zkratek a symbolů

A

a_1 – časová konstanta SOTD modelu
a_2 – časová konstanta SOTD modelu

B

$button_{IsPresed}$ – proměnná ukazující na stisknutí tlačítka start

C

CFC – jazyk plovoucích schémat (Continuous flow chart)
CIB – sběrnice Common installation bus
$currentTime$ – aktuální čas

D

d_1 – odchylka akční veličiny v horní polovině relé od pracovního bodu
d_2 – odchylka akční veličiny ve spodní polovině relé od pracovního bodu
$divisionT_dT$ – podíl dopravního zpoždění a časové konstanty

E

e – regulační odchylka
$exponentialTimeSpan$ – čas od počátku exponenciálního růstu akční veličiny

F

FBD – jazyk funkčního blokového schématu (Function block diagram)
fbFirstOrder – funkční blok ro simulaci soustavy prvního řádu

fbstabilization – funkční blok pro kontrolu ustálení soustavy
fbWebInputControl – funkční blok pro kontrolu správnosti zadaných parametrů regulace
FOTD – model prvního řádu s dopravním zpožděním (First order time delayed model)

G

G – statická citlivost simulované soustavy
G_1 – bod Nyquistovy frekvenční charakteristiky s frekvencí ω_1
G_2 – bod Nyquistovy frekvenční charakteristiky s frekvencí ω_2
$G(j\omega_1)$ – bod Nyquistovy frekvenční charakteristiky s frekvencí ω_1
$G(j\omega_2)$ – bod Nyquistovy frekvenční charakteristiky s frekvencí ω_2
$G(j\omega_u)$ – bod Nyquistovy frekvenční charakteristiky s frekvencí ω_u
$G(j\omega_{u2})$ – bod Nyquistovy frekvenční charakteristiky s frekvencí ω_{u2}
$G(s)$ – Laplaceův přenos modelu soustavy

H

h – mez hystereze relé
H – rozsah hodnot výstupu relé
h_1 – horní mez hystereze relé
h_2 – dolní mez hystereze rélé
$hysteresis_{HighValue}$ – mez hystereze rélé nad pracovním bodem
$hysteresis_{LowValue}$ – mez hystereze relé pod pracovním bodem

I

IFOTD – model prvního řádu s integrací s dopravním zpožděním (Integrating plus first order time delayed model)
IL – jazyk seznamu instrukcí (Instruction list)
Im(G(jω)) – imaginární souřadnice bodu Nyguistovy frekvenční charakteristiky
$indexIntegral$ – index pole při integraci
$indexShiftIntegral$ – index pole při integraci posunutý o půl periody
$integralExponent1$ – exponent v integraci při výpočtu bodu s frekvencí identifikace
$integralExponent2$ – exponent v integraci při výpočtu bodu s dvojnásobnou frekvencí vzhledem k frekvenci identifikace
$integral_{U1}$ – výsledek integrálu z akční veličiny při frekvenci integrace
$integral_{U2}$ – výsledek integrálu z akční veličiny při dvojnásobku frekvence integrace
$integral_{Y1}$ – výsledek integrálu z akční veličiny při frekvenci integrace
$integral_{Y2}$ – výsledek integrálu z akční veličiny při dvojnásobku frekvence integrace
ITD – integrační model s dopravním zpožděním (Integrating time delayed model)
I_u – integrál z akční veličiny přes jednu periodu kmitu
I_y – integrál z regulované veličiny přes jednu periodu kmitu

K

K – statická citlicost soustavy

L

LCD – displej z tekutých krystalů (Liquid crystal display)
LD – jazyk příčkového diagramu (Ladder diagram)
$leadingEdgeCounter$ – čítač náběžných hran
$leadingEdgeRTC$ – čas náběžné hrany
$leadingEdgeRtcTemp$ – pomocný čas náběžné hrany

M

MMC – označení pro typ paměťové karty (Multimedia card)

N

$noise_{Max}$ – maximální absolutní hodnota šumu

P

PD – proporcionálně derivační (regulátor)
PI – proporcionálně integrační (regulátor)
PID – proporcionálně integračně derivační (regulátor)
PLC – programovatelný automat (Programmable logic controller)

R

RISC – architektura procesorů s omezenou instrukční sadou (Reduced instruction set computer)
RTC – hodiny reálného času (Real time clock)
Re(G(jω)) – reálná souřadnice bodu Nyguistovy frekvenční charakteristiky

S

s – Laplaceův operátor
$samplingArrayColumnIndex$ – index sloupce, kam se mají ukládat data z aktuální periody
$samplingArrayColumnNeighbour1Index$ – index sloupce, kam se ukládala data v jedné ze dvou předcházejícíc period
$samplingArrayColumnNeighbour2Index$ – index sloupce, kam se ukládala data

v jedné ze dvou předcházejícíc period
samplingArrayIndex – index řádku ve sloupci pole, kam se má uložit vzorek
samplingArrayLastIndex – index řádku ve sloupci pole, kam se uložil poslední vzorek
samplingArrayLastIndexes – počet uložených vzorků
samplingPeriodTimeSpan – vzorkovací perioda
SD – označení pro typ paměťových karet (Secure Digital)
SFC – jazyk sekvenčního přechodového diagramu (Sequential function chart)
SISO – systém typu jeden vstup – jeden výstup (Single input single output)
SOTD – model druhého řádu s dopravním zpožděním (Second order time delayed model)
$SOTD_{A1}$ – časová konstanta SOTD modelu
$SOTD_{A2}$ – časová konstanta SOTD modelu
$SOTD_K$ – statická citlivost SOTD modelu
$SOTD_\tau$ – dopravní zpoždění SOTD modelu
ST – jazyk strukturovaného textu (Structured text)
Start – tlačítko pro spuštění identifikace
$Start_{ExponentialTime}$ – čas počátku exponenciálního růstu akční veličiny

T

T časová konstanta modelu prvního řádu s dopravním zpožděním, jinde vzorkování simulované soustavy
tauIndexIntegral – čas při integraci
t_1 – čas, po který je relé v horní poloze
t_2 – čas, po který je relé ve spodní poloze
T_1 – čas, po který je relé v horní poloze
T_{1Temp} – pomocný čas, po který je relé v horní poloze
T_2 – čas, po který je relé ve spodní poloze
T_{2Temp} – pomocný čas, po který je relé ve spodní poloze
tempBool – dočasná hodnota typu BOOL
*tempBool*2 – dočasná hodnota typu BOOL
tempCounter – dočasný čítač
tempDataTime – dočasná hodnota typu DATE AND TIME
tempReal – dočasná hodnota typu REAL
tempSum – suma pro výpočet průměrné hodnoty regulované veličiny
T_d – dopravní zpoždění
TimerForDelay – časovač pro zpoždění spádové a náběžné hrany
t_{off} – čas po který je relé ve spodní poloze
t_{on} – čas po který je relé v dolní poloze
t_p – perioda kmitu
T_{period} – perioda kmitu
traffic Delay – zpoždění spádové a náběžné hrany
trailingEdgeCounter – čítač spádových hran
trailingEdgeRTC – čas spádové hrany
trailingEdgeRtcTemp – pomocný čas spádové hrany

T_u – dopravní zpoždění vypočtené z bodu $G(j\omega_u)$
T_{u2} – dopravní zpoždění vypočtené z bodu $G(j\omega_{u2})$
t_{uy} – časový posuv mezi akční a regulovanou veličinou

U

u – akční veličina
u_0 – hodnota akční veličiny v pracovním bodě
u_1 – hodnota akční veličiny při horní poloze relé
u_2 – hodnota akční veličiny při spodní poloze relé
$u_a(t)$ – průběh akční veličiny při dvojnásobné frekvenci identifikace
u_{K1} – hodnota akční veličiny v bodě blízkém pracovnímu bodu
u_{K2} – hodnota akční veličiny v bodě blízkém pracovnímu bodu potažmo v pracovním bodě
$u_{MaxValue}$ – horní mezní hodnota akční veličiny
$u_{MinValue}$ – spodní mezní hodnota akční veličiny
u_{off} – hodnota akční veličiny při spodní poloze relé
u_{on} – hodnota akční veličiny při horní poloze relé
$u_{sampleComplex1}$ – vzorek akční veličiny ve formátu komplexního čsla pro výpočet bodu frekvenční charakteristiky s frekvencí identifikace
$u_{sampleComplex2}$ – vzorek akční veličiny ve formátu komplexního čsla pro výpočet bodu frekvenční charakteristiky s frekvencí dvojnásobnou k frekvenci identifikace
$u_{samplingArray}$ – pole pro ukládání vzorků akční veličiny ve třech po sobě následujících periodách
$u(t)$ – hodnota akční veličiny v čase
$u(t^{-1})$ – hodnota akční veličiny v předchozím čase newline $u_{ValueTemp}$ – dočasná hodnota akční veličiny při určování parametrů relé newline $u_{ValueTemp2}$ – dočasná hodnota akční veličiny při určování parametrů relé

W

w – žádaná veličina

x

x – koeficient pro výpočet velikosti zpoždění spádové a náběžné hrany

Y

y – regulovaná veličina
y_0 – hodnota regulované veličiny v pracovním bodě
y_A – amplituda kmitů regulované veličiny
$y_a(t)$ – průběh regulované veličiny při dvojnásobné frekvenci identifikace

y_{K1} – hodnota akční veličiny v bodě blízkém pracovnímu bodu
y_{K2} – hodnota akční veličiny v bodě blízkém pracovnímu bodu potažmo v pracovním bodě
$y_{MaxHighDeviation}$ – maximální výchylka regulované veličiny nad pracovní bod
$y_{MaxLowDeviation}$ – maximální výchylka regulované veličiny pod pracovní bod
$y_{sampleComplex1}$ – vzorek regulované veličiny ve formátu komplexního čsla pro výpočet bodu frekvenční charakteristiky s frekvencí identifikace
$y_{sampleComplex2}$ – vzorek regulované veličiny ve formátu komplexního čsla pro výpočet bodu frekvenční charakteristiky s frekvencí dvojnásobnou k frekvenci identifikace
$y_{samplingArray}$ – pole pro ukládání vzorků regulované veličiny ve třech po sobě následujících periodách
$y_{samplingArrayDiffA}$ – rozdíl mezi vzorky regulované veličiny ve dvou periodách
$y_{samplingArrayDiffB}$ – rozdíl mezi vzorky regulované veličiny ve dvou periodách
$y_{samplingArrayDiff01}$ – rozdíl mezi vzorky regulované veličiny ve dvou periodách
$y_{samplingArrayDiff02}$ – rozdíl mezi vzorky regulované veličiny ve dvou periodách
$y_{samplingArrayDiff12}$ – rozdíl mezi vzorky regulované veličiny ve dvou periodách
$y(t)$ – hodnota regulované veličiny v čase

γ

γ – stupeň asymetrie relé

ϕ

ϕ – fázový posuv [rad]

ρ

ρ – index poloviny periody
ρ_{Temp} – pomocný index poloviny periody
ρ_{uy} – fázový posuv mezi akční a regulovanou veličinou [rad]

τ

τ – normalizované dopravní zpoždění
τ_1 – časová konstanta anisochronního modelu [s]
τ_2 – časová konstanta anisochronního modelu [s]
τ_y – zpoždění hlavní zpětné vazby anisochronního modelu [s]

ω

ω – úhlová frekvence [rad]
ω_0 – nulová frekvence [rad/s]

ω_1 – frekvence identifikace [rad/s]
ω_2 – dvojnásobek frekvence identifikace [rad/s]
ω_u – frekvence identifikace [rad/s]
ω_{u2} – dvojnásobek frekvence identifikace [rad/s]

Příloha C

Skripty reléové identifikace a funkční bloky

Skripty reléové identifikace a funkční bloky jsou přiloženy v tomto pořadí:

Určení parametrů modelu ze tří bodů Nyquistovy frekvenční charakteristiky
Určení parametrů modelu ze tří bodů Nyquistovy frekvenční charakteristiky s přidaným dopravním zpožděním
Určení modelu pro nastavení parametrů PI, PD a PID regulátorů
fbwebInputValidation
fbstabilization
fbparametrySOTD
fbRelayFeedbackIdentification

```
// Urceni parametru modelu ze tri bodu Nyquistovy frekvencni charakteristiky

VAR_GLOBAL RETAIN
  ySamplingArray : ARRAY [0..2, 0..1300] OF REAL; // vzorkovani prubehu
regulovane veliciny
  uSamplingArray : ARRAY [0..2, 0..1300] OF REAL; //vzorkovani prubehu akcni
veliciny
  samplingArrayLastIndexes : ARRAY [0..2] OF UINT;  //pocty vzorku
  samplingArrayColumnIndex   : USINT := 2; //index používaného sloupce
  yK1 : REAL; // regulovana veličina v bode blizkem pracovnímu bodu
  yK2 : REAL; // regulovana veličina v bode blizkem pracovnímu bodu
  uK1 : REAL; // akcni veličina v bode blizkem pracovnímu bodu
  uK2 : REAL; // akcni veličina v bode blizkem pracovnímu bodu
  trailingEdgeIndex : ARRAY [0..2] OF UINT;  //index na kterem doslo k prepnuti
rele (u)
  currentLastIndex : UINT; // minuly index
END_VAR

VAR_GLOBAL CONSTANT
  BufferArrayLength : UINT := 1200;
END_VAR
VAR_GLOBAL
//vetrak + prutokomer
(*
  prutokomer AT r0_p3_AI1.ENG : REAL; //vstup do PLC
  ventilator AT r0_p3_AO1.ENG : REAL; //vystup z PLC
  u AT ventilator : REAL; //akcni zasah
  y_puvodni AT prutokomer; //vystup soustavy
*)
//u : REAL :=0;
//y : REAL;
//zarovka + termistor
//(*
  termistor AT r0_p3_AI2.ENG : REAL; //vstup do PLC
  zarovka   AT r0_p3_AO0.ENG : REAL; //vystup z PLC
  u AT zarovka : REAL; //akcni zasah
  y AT termistor; //vystup soustavy
//*)
//levitace
(*
  prutokomer AT r0_p3_AI1.ENG : REAL; //vstup do PLC
  ventilator AT r0_p3_AO1.ENG : REAL; //vystup z PLC
  u AT ventilator : REAL; //akcni zasah
  y_puvodni AT prutokomer; //vystup soustavy
  y:REAL;
*)
END_VAR

PROGRAM prgMain
  VAR_INPUT
  END_VAR
  VAR_OUTPUT
  END_VAR
  VAR
  leadingEdgeCounter : INT;  // pocet nabeznych hran
  runningBlockCase : INT :=0;  // aktualni blok kodu
  tempCounter : INT :=0;  // pocet vzorku po detekci ustaleni (yk1, yk2)
  tempCounter2 : INT :=0;  // citac
  trailingEdgeCounter : INT;  // pocet sestupnych hran
  indexIntegral   : UINT;  // index pri integraci
  indexShiftIntegral : UINT;  // posunuty index pri integraci (posuv = Tperiod/2)
  samplingArrayColumnNeighbour1Index : UINT;   // index sousedního sloupce pole
  samplingArrayColumnNeighbour2Index : UINT;    // index sousedního sloupce pole
  samplingArrayIndex   : UINT;   // index pri vzorkovani
  samplingArrayIndexOld : UINT;  // puvodni index pri vzorkovani z predchoziho
pruchodu
  buttonIsPressed : BOOL :=1;   // input z buttonu
  tempBool : BOOL := false;  // docasna bool hodnota
```

```
  tempBool2 : BOOL := false;  // docasna bool hodnota
  samplingExit : bool := false;   // priznak, zda opustit vzorkovani
  hysteresisHighValue : REAL;  // horni mez hystereze
  hysteresisLowValue : REAL;  // dolni mez hystereze
  integralExponent1 : REAL;  // exponent ve vzorci s integralem (realny)
  integralExponent2 : REAL;  // exponent ve druhem vzorci s integralem (realny)
  K :REAL;  // staticka citlivost
  noiseMax : REAL;  // maximalni sum
  omega1 : REAL;   // uhlova rychlost ve vzorci s integralem
  omega2 : REAL;   // uhlova rychlost ve druhem vzorci s integralem
  samplingDelayMax : REAL; // maximalni zpozdeni pri vzorkovani pole (0;
samplingPeriodTimeSpan)
  samplingDelaySum : REAL;  // suma vsech zpozdeni pri vzorkovani pole
  samplingDelaySumMax : REAL;  // maximalni suma vsech zpozdeni pri samplovani
poli
  samplingPeriodTimeSpan :  REAL;   // vzorkovací perioda integrátoru
  sotdA1 : REAL;   // casova konstanta modelu SOTD
  sotdA2 : REAL;   // druha casova konstanta modelu SOTD
  sotdK : REAL;   // staticka citlivost v modelu SOTD
  sotdTau :REAL;   // dopravni zpozdeni v modelu SOTD
  tauIndexIntegral : REAL;   // cas pri integraci
  tempReal : REAL;  // docasna promenna pro real
  tempReal4 : REAL;   // docasna promenna pro real
  tempSum : REAL;   // pomocna promenna pro vypocet prumeru vystupu
  timeDiff : REAL;    // pomocna promenna pro urcovani rozdilu casu
  trafficDelay : REAL;   // dopravni zpozdeni
  T1 : REAL;   // cas pro rele v horni poloze
  T2 : REAL;   // cas pro rele ve spodni poloze
  Tperiod : REAL;   // delka periody [ms]
  u0 : REAL := 3;    // vstup v pracovnim bode
  uHighValue : REAL;  // hodnota vstupu pri horni poloze rele
  uLowValue : REAL;   // hodnota vstupu pri spodni poloze rele
  uMaxValue : REAL := 10;  // maximalni hodnota vstupu
  uMinValue : REAL := 0;  // minimalni hodnota vstupu
  y0 : REAL;  // vystup v pracovnim bode
  ySamplingArrayDiffSum : REAL; // suma rozdilu mezi namerenymi hodnotami Y v
poslednich trech pruchodech
  ySamplingArraySum : REAl;  // suma namerenych hodnot v poli Y
  G1  : COMPLEX;  // prvni bod Nyquistovy frekvencni charakteristiky v komplexni
rovine
  G2  : COMPLEX;  // druhy bod Nyquistovy frekvencni charakteristiky v komplexni
rovine
  integralComplexExponent1 : COMPLEX;  // komplexni exponent v prvnim integralu
  integralComplexExponent2 : COMPLEX;  // komplexni exponent ve druhem integralu
  integralU1 : COMPLEX;  // vysledek celeho integralu ve jmenovateli 1. vzorce
  integralU2 : COMPLEX;  // vysledek celeho integralu ve jmenovateli 2. vzorce
  integralY1 : COMPLEX;  // vysledek celeho integralu v citateli 1. vzorce
  integralY2 : COMPLEX;  // vysledek celeho integralu v citateli 2. vzorce
  uSampleComplex1 : COMPLEX;  // vzorek v komplexni rovine ve jmenovateli 1.
vzorce
  uSampleComplex2 : COMPLEX;  // vzorek v komplexni rovine ve jmenovateli 2.
vzorce
  ySampleComplex1 : COMPLEX;  // vzorek v komplexni rovine v citateli 1. vzorce
  ySampleComplex2 : COMPLEX;  // vzorek v komplexni rovine v citateli 2. vzorce
  leadingEdgeRtc : DATE_AND_TIME;  // cas nabezne hrany
  trailingEdgeRtc : DATE_AND_TIME;  // cas sestupne hrany
  tempDateTime : DATE_AND_TIME;  // docasna promenna pro datetime
  T0 : DATE_AND_TIME;  // pocatek vzorkovaci periody
  timer : TON;  // casovac
  timerForDelay : TON;  //casovac pro zpozdeni nabezne a spadove hrany
  webOutputString : STRING;  // vystup na web
  fb_webInputValidation  :  webInputValidation;  //funkci  blok  pro  kontrolu
zadanych udaju
  fb_stabilization : fbstabilization;  //funkcni blok pro kontrolu ustaleni
  fb_parametrySOTD : fbparametrySOTD;  //funkcni blok vypoctu parametru modelu
  END_VAR
  VAR_TEMP
  END_VAR
```

```
  CASE runningBlockCase OF

////////////////////////////////////////////////////////////////////////////
////////   nastaveni   parametru   a   cekani   na   buttonIsDisabled   ////////
////////////////////////////////////////////////////////////////////////////

  00: // Init
    runningBlockCase := 01; // prechod do dalsiho CASE

  01: // Main
    IF buttonIsPressed = TRUE THEN // kontrola stisknuti tlacitka
      fb_webInputValidation(minValue:= uMinValue, minOffset := -0.5, maxValue:=
uMaxValue,  value:=u0,  isValid  =>  tempBool,  message  =>  webOutputString);
//volání podprogramnu pro kontrolu zadaných parametru
      IF tempBool = FALSE THEN // kdyz jsou spatne zadane udaje
        buttonIsPressed := FALSE; //uvolněni tlacitka
      ELSE
        runningBlockCase := 10; // prechod do dalsiho CASE
      END_IF;
    END_IF;

////////////////////////////////////////////////////////////////////////////
////////////////////////////  cekani na ustaleni  //////////////////////////
////////////////////////////////////////////////////////////////////////////

  10: // Init
    u:=u0-0.5;    // nastavení vstupního napití pod pracovní bod, kvuli ureení
statické citlivosti
    runningBlockCase := 11; // prechod do dalsiho CASE

  11: // Main
    fb_stabilization(vstup:=y,  cas_mereni:=  T#10s,  ustaleno  =>  tempBool);  //
cekani na ustaleni vystupu
    IF tempBool = TRUE THEN // kdyz je vystup ustaleny
      runningBlockCase := 20; // prechod do dalsiho CASE
    END_IF;

////////////////////////////////////////////////////////////////////////////
////////////////////////////  urceni u1 a y1  //////////////////////////////
////////////////////////////////////////////////////////////////////////////

  20: // Init
    tempCounter := 0; //vynulovani citace
    tempSum := 0; //vynulovani promenne
    runningBlockCase := 21; // prechod do dalsiho CASE

  21: // Main
    timer(IN:=TRUE, PT:=T#1s); // casovac pro urceni prumerne hodnoty vystupu v
bode 1 pro urceni staticke citlivosti
    tempSum := tempSum+y;  // suma vystupu po dobu behu casovace
    tempCounter := tempCounter+1; // pocet sectenych vystupu po dobu casovace
    IF timer.Q THEN // po uplynutí easovaee
      timer(IN:=FALSE); // vynulování casovace
      yK1  :=  tempSum/INT_TO_REAL(tempCounter);  //  vypocet  prumerne  hodnoty
vystupu
      uK1 := u0 - 0.5; // hodnota akcni veliciny
      runningBlockCase := 30; // prechod do dalsiho CASE
    END_IF;

////////////////////////////////////////////////////////////////////////////
////////////////////////////// cekani na ustaleni  //////////////////////////
////////////////////////////////////////////////////////////////////////////

  30: // Init
    u:=u0; //nastavení akcni veliciny na zadanou hodnotu, tj. pracovni bod
    runningBlockCase := 31; // prechod do dalsiho CASE
```

```
  31: // Main
    fb_stabilization(vstup:=y, cas_mereni:= T#10s, ustaleno => tempBool); //
cekani na ustaleni vystupu
    IF tempBool = TRUE THEN // kdyz je ustaleny vystup
       runningBlockCase := 40; // prechod do dalsi faze programu
    END_IF;

////////////////////////////////////////////////////////////////////////////
/////////////////////////// urceni u2 a y2  ////////////////////////////////
////////////////////////////////////////////////////////////////////////////


  40: // Init
    tempSum := 0; // vynulovani promenne
    tempCounter := 0; // vynulovyni citace
    runningBlockCase := 41; // prechod do dalsiho CASE

  41: // Main
    timer(IN:=TRUE, PT:=T#1s); // casovac pro vypocteni prumerne hodnoty vystupu
    tempSum := tempSum+y; // suma vzorku vystupu
    tempCounter := tempCounter+1; //pocet sectenych vystupu
    IF timer.Q THEN // když casovac dobehne
      timer(IN:=FALSE); // nulovani casovace
      yK2 := tempSum/INT_TO_REAL(tempCounter); // vypocet prumerneho vystupu v
bode pro urceni staticke citlivosti
      uK2 := u0; //akcni velicina v bode pro urceni staticke citlivosti
      y0:=yK2; //regulovana velicina v pracovnim bode
      K:=(yK2-yK1)/(uK2-uK1); //vypocet hodnoty statické citlivosti
      runningBlockCase:=50; // prechod do dalsiho CASE
    END_IF;

////////////////////////////////////////////////////////////////////////////
////////////////////////////////// mereni sumu ////////////////////////////////
////////////////////////////////////////////////////////////////////////////

  50: // Init
    noiseMax := 0; //vynulování nejvyšší hodnoty výstupu
    runningBlockCase:=51; // prechod do dalsiho CASE

  51: // Main
    timer(IN:=TRUE, PT:=T#10s); // casovac po ktery se meri sum
    IF timer.Q = FALSE THEN // kdyz casovac nedobehl
      IF ABS(y0-y)>noiseMax THEN // kdyz absolutní hodnota sumu je vetsi nez
nejvetsi ulozena
        noiseMax:=ABS(y0-y); // ulozeni velikosti sumu z aktualni absoulutni
hodnoty sumu
      END_IF;
    ELSE
      timer(IN:=FALSE); // vynulovani casovace
      hysteresisHighValue:=noiseMax*8; // urceni mezi hystereze rele z velikosti
sumu
      hysteresisLowValue:=noiseMax*10; // urceni mezi hystereze rele z velikosti
sumu
      runningBlockCase:=70; // prechod do dalsiho CASE
    END_IF;

  (* // uprava pro pouziti simulace soustavy
    noiseMax:=0.2;
    hysteresisHighValue:=noiseMax*6;
    hysteresisLowValue:=noiseMax*10;
    runningBlockCase :=70;
  *)

////////////////////////////////////////////////////////////////////////////
//////////////////////////////  urceni periody  ////////////////////////////////
////////////////////////////////////////////////////////////////////////////

  70: // Init
```

```
    uHighValue := u0 + 5; // urceni horni polohy rele
    IF uHighValue > uMaxValue THEN // pokud horni poloha rele je vyssi nez
maximalni
      uHighValue:= uMaxValue; // uprava horni hodnoty rele
    END_IF; // horni mez rele

    uLowValue:= u0 - 3; // dolní poloha rele
    IF uLowValue < uMinValue THEN // kdyz je spodni poloha relé mensi nez
minimalni
      uLowValue:= uMinValue; // uprava spodni polohy rele
    END_IF;
    u:= uHighValue; // nastaveni akcni veliciny na horni mez rele
    trailingEdgeCounter:=0; // citac spadovych hran
    leadingEdgeCounter:=1; // citac nabeznych hran
    tempBool := FALSE; // ukazatel stejne delky casu T1
    tempBool2 := FALSE; // ukazatel stejne delky casu T2
    T1 := 1; // nastaveni casu T1 na nenulovou hodnotu
    T2 := 1; // nastaveni casu T2 na nenulovou hodnotu
    leadingEdgeRtc := getRTC(); // cas nabezne hrany
    trailingEdgeRtc := getRTC(); // cas spadove hrany
    runningBlockCase := 71; // prechod do dalsiho CASE

  71: // Main
  IF y>y0+hysteresisHighValue AND u=uHighValue THEN // kdyz je cas na sestupnou
hranu a jeste k ni nedoslo
    u:= uLowValue; // dolni mez rale
    tempDateTime:= getRTC(); // ulozeni aktualniho casu
    trailingEdgeCounter := trailingEdgeCounter + 1; // citac spadovych hran
    trailingEdgeRtc := tempDateTime; // cas spadove hrany
    tempReal  :=  (TIME_TO_REAL(SUB_DT_DT(tempDateTime,leadingEdgeRtc)));  //
aktualni cas T1
    IF abs(1.0 - tempReal/T1) < 0.03 THEN // kdyz je rozdil mezi soucasnym a
minulym T1 rozdil mensi nez 3 procenta
      tempBool := TRUE; // ukazatel splneni podminky
    ELSE
      tempBool := FALSE; // ukazatel na nesplneni podminky
    END_IF;
    T1 := tempReal; // prepsani casu T1
  END_IF;
  IF y<y0-hysteresisLowValue AND u=uLowValue THEN // kdyz ma byt nabezna hrana
a jeste k ni nedoslo
    u:= uHighValue; // nastavi se nabezna hrana
    tempDateTime:= getRTC(); // cas nabezne hrany
    leadingEdgeCounter := leadingEdgeCounter + 1; // citac nabeznych hran
    leadingEdgeRtc := tempDateTime; // cas nabezne hrany
    tempReal  :=  (TIME_TO_REAL(SUB_DT_DT(tempDateTime,trailingEdgeRtc)));  //
vypocet casu T2
    IF abs(1.0 - tempReal/T2) < 0.03 THEN // kdyz je rozdil mezi soucasnym a
minulym T2 rozdil mensi nez 3 procenta
      tempBool2 := TRUE; // ukazatel splneni podminky
    ELSE
      tempBool2 := FALSE; // ukazatel nesplneni podminky
    END_IF;
    T2 := tempReal; // cas T2
  END_IF;
  IF tempBool AND tempBool2 AND u=uHighValue THEN // kdyz jsou splneny podminky
a rele je nahore
    Tperiod := T1+T2; // vypocet periody
    runningBlockCase :=80; // prechod do dalsiho CASE
  END_IF;

////////////////////////////////////////////////////////////////////////////
//////////////////////////////  vzorkovani  ////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

  80: // Init
    IF T1 > T2 THEN // kontrola splneni podminky
```

```
      webOutputString := 'T1 je vetsi nez T2, sprav si to';   // hlaseni o
nesplneni podminky
      runningBlockCase := 1000; // prechod do dalsiho CASE
    END_IF;
    samplingArrayIndexOld := 1000; // nastaveni indexu na nesmyslny
    samplingArrayColumnIndex := 0; // nastaveni sloupce pole
    samplingArrayColumnNeighbour1Index := 1; // nastaveni sousedniho sloupce
    samplingArrayColumnNeighbour2Index := 2; // nastaveni druheho sousedniho
sloupce
    ySamplingArrayDiffSum := 0; // nulovani souctu rozdilu mezi sloupci
    ySamplingArraySum := 0; // soucet ulozenych prvku
    samplingArrayLastIndexes[0] := 1000; // nastaveni indexu na nesmyslny
    samplingArrayLastIndexes[1] := 1000; // nastaveni indexu na nesmyslny
    samplingArrayLastIndexes[2] := 1000; // nastaveni indexu na nesmyslny
    tempCounter := 0; // citac
    tempCounter2 := 0; // citac
    samplingDelayMax := 0; // nulovani maximalniho zpozdeni
    samplingDelaySum := 0; // nulovani souctu zpozdeni
    samplingDelaySumMax := 0; // nulovani maximalniho celkoveho souctu zpozdeni
    samplingPeriodTimeSpan := Tperiod / 900.0; // vzorkovaci perioda [ms], ale
opravdu! [ms]!!
    samplingPeriodTimeSpan  :=  10*CEIL(samplingPeriodTimeSpan/10);  //  uprava
vzorkovaci periody
    IF (samplingPeriodTimeSpan < 20) THEN  // kdyz je vzorkovaci perioda mensi
nez 20ms
      samplingPeriodTimeSpan := 20; // vzorkovaci perioda 20ms
    END_IF;
    T0:=GetRTC(); // aktualni cas
    runningBlockCase := 81; // prechod do dalsiho CASE

  81: // Main
    IF y>y0+hysteresisHighValue AND u=uHighValue THEN // kdyz je cas na sestupnou
hranu
      timerForDelay(IN := TRUE, PT := T#0s); // zpozdeni sestupne hrany
      IF timerForDelay.Q = TRUE THEN  // kdyz casovac dobehne
        u:= uLowValue; // sestupna hrana
        tempCounter2 := tempCounter2+1; //citac
        timerForDelay(IN := FALSE); // vynulovani casovace
        tempDateTime:= getRTC();  // aktualni cas
        trailingEdgeRtc := tempDateTime; // cas spadove hrany
        tempReal  :=  (TIME_TO_REAL(SUB_DT_DT(tempDateTime,leadingEdgeRtc)));  //
T1
        T1 := tempReal; // T1
      END_IF;
    END_IF;
    IF y<y0-hysteresisLowValue AND u=uLowValue THEN // cas na nabeznou hranu
      timerForDelay(IN := TRUE, PT := T#0s ); // zpozdeni nabezne hrany
      IF timerForDelay.Q = TRUE THEN // dobehl casovac
        u:= uHighValue; // nabezna hrana
        timerForDelay(IN := FALSE); // vynulovani casovace
        tempDateTime:= getRTC(); // aktualni cas
        leadingEdgeRtc := tempDateTime; //cas nabezne hrany
        tempReal := (TIME_TO_REAL(SUB_DT_DT(tempDateTime,trailingEdgeRtc))); //
T2
        T2 := tempReal; // T2
        T0:= getRTC(); // aktualni cas
        IF (samplingDelaySum > samplingDelaySumMax) THEN // kontrola hodnoty
sumy zpozdeni
          samplingDelaySumMax  :=  samplingDelaySum;  //  prepsani  hodnoty  sumy
zpozdeni
        END_IF;
        samplingDelaySum := 0; // vynulovani sumy zpozdeni
        IF tempCounter2 > 3 AND ySamplingArrayDiffSum/ySamplingArraySum < 0.06
THEN  // kontrola opousteci podminky
          samplingExit := TRUE; // ukazatel ukonceni vzorovani
          runningBlockCase := 90; // prechod do dalsiho CASE
        ELSE
```

```
          samplingArrayColumnIndex := samplingArrayColumnIndex + 1; // prechod
do dalsiho sloupce
        IF samplingArrayColumnIndex > 2 THEN  // kontrola cisla sloupce
          samplingArrayColumnIndex := 0; // uprava cisla sloupce
        END_IF;
        samplingArrayLastIndexes[samplingArrayColumnIndex] := 0; // vynulovani
cisla posledniho vzorku
        CASE samplingArrayColumnIndex OF  // upravy sloupcu sousedu
        0 :
          samplingArrayColumnNeighbour1Index := 1;
          samplingArrayColumnNeighbour2Index := 2;
        1 :
          samplingArrayColumnNeighbour1Index := 0;
          samplingArrayColumnNeighbour2Index := 2;
        2:
          samplingArrayColumnNeighbour1Index := 0;
          samplingArrayColumnNeighbour2Index := 1;
        END_CASE;
        ySamplingArraySum := 0; // vynulovani souctu vzorku
        ySamplingArrayDiffSum := 0; // vynulovani souctu rozdilu vzorku
      END_IF;
    END_IF;
    END_IF;
    IF samplingExit = FALSE THEN  // kdyz neni ukonceno vzorkovani
      timeDiff  :=  TIME_TO_REAL(SUB_DT_DT(GetRTC(),T0));  //  cas  od  pocatku
periody
      samplingArrayIndex        :=        REAL_TO_UINT(FLOOR(timeDiff        /
samplingPeriodTimeSpan)); // index vzorku
      IF (samplingArrayIndex <> samplingArrayIndexOld) THEN // kdyz mame novy
index vzorku
        ySamplingArray[samplingArrayColumnIndex,samplingArrayIndex]  :=  y;  //
ulozeni vzorku regulovane veliciny
        uSamplingArray[samplingArrayColumnIndex,samplingArrayIndex]  :=  u;  //
ulozeni vzorku akcni veliciny
        samplingArrayLastIndexes[samplingArrayColumnIndex]:=
samplingArrayIndex; // prepsani posledniho indexu
        IF   samplingArrayLastIndexes[samplingArrayColumnNeighbour1Index]   >=
samplingArrayIndex THEN // pokud nebyl presazen pocet vzorku v sousednim sloupci
          ySamplingArrayDiffSum    :=    ySamplingArrayDiffSum    +    abs(y-
ySamplingArray[samplingArrayColumnNeighbour1Index   ,samplingArrayIndex]);   //
prepocet sumy rozdilu
        END_IF;
        IF   samplingArrayLastIndexes[samplingArrayColumnNeighbour2Index]   >=
samplingArrayIndex THEN // pokud nebyl presazen pocet vzorku v sousednim sloupci
          ySamplingArrayDiffSum    :=    ySamplingArrayDiffSum    +    abs(y-
ySamplingArray[samplingArrayColumnNeighbour2Index   ,samplingArrayIndex]);   //
prepocet sumy rozdilu
        END_IF;
        ySamplingArraySum := ySamplingArraySum + abs(y); // suma ulozenych prvku
        IF samplingArrayIndex = 0 THEN // kdyz je pocatek periody
          tempReal4 := timeDiff; // rozdil casu
        ELSE
          tempReal4  :=  timeDiff  -  (UINT_TO_REAL(samplingArrayIndexOld  +  1)  *
samplingPeriodTimeSpan); // rozdil casu
        END_IF;
        IF (tempReal4 > samplingDelayMax) THEN  // kontrola maximalniho zpozdeni
          samplingDelayMax := tempReal4; // maximalni zpozdeni
        END_IF;
        samplingDelaySum  :=  samplingDelaySum  +  tempReal4;  //  suma  zpozdeni
vzorkovani
        IF    (samplingArrayIndex    <>    0    AND    samplingArrayIndex    <>
samplingArrayIndexOld + 1) THEN
          tempCounter := tempCounter + 1; // citac vzorku
        END_IF;
        samplingArrayIndexOld := samplingArrayIndex; // minuly index
      END_IF;
    END_IF;
```

```
////////////////////////////////////////////////////////////////////////////
//////////////////////////// vypocet integralu ////////////////////////////
////////////////////////////////////////////////////////////////////////////

  90: // Init
    Tperiod := T1+ T2; // vypocet periody
    omega1   := MATH.PI*2 / (Tperiod * 0.001); // frekvence identifikace
    omega2  := 2.0*omega1; // dvojnasobek frekvence identifikace
    indexIntegral:=0; // index prvku pro integraci
    indexShiftIntegral:=
(samplingArrayLastIndexes[samplingArrayColumnIndex]/2); // posunuty index prvku
pro integraci
    currentLastIndex:=  samplingArrayLastIndexes[samplingArrayColumnIndex];  //
posledni index vzorku
    u :=uMinValue; // minimalizovani hodnoty akcni veliciny
    runningBlockCase := 91; // prechod do dalsiho CASE

  91: // Main
      IF  indexShiftIntegral  >  currentLastIndex  THEN  //  kontrola  posunuteho
indexu v integralu
        indexShiftIntegral := indexShiftIntegral - currentLastIndex; // uprava
posunuteho indexu v integralu
      END_IF;
      tauIndexIntegral    := UINT_TO_REAL(indexIntegral)*samplingPeriodTimeSpan
/ 1000; // cas v integralu jako desetiné císlo v sekundách
      integralExponent1:= omega1*tauIndexIntegral; // exponent integralu
      integralExponent2:= omega2*tauIndexIntegral; // exponent integralu
      integralComplexExponent1.re := 0; // realna slozka exponentu v integralu
      integralComplexExponent1.im := -integralExponent1; // imaginarni slozka
exponentu v integralu
      uSampleComplex1.im:=0; // imaginarni slozka vzorku akci veliciny
      uSampleComplex1.re:=
(uSamplingArray[samplingArrayColumnIndex,indexIntegral]);   //   realna   slozka
vzorku akci veliciny
      ySampleComplex1.im:=0; // imaginarni slozka vzorku regulovane veliciny
      ySampleComplex1.re:=
(ySamplingArray[samplingArrayColumnIndex,indexIntegral]);   //   realna   slozka
vzorku regulovane veliciny
      integralComplexExponent2.re := 0; // relna slozka exponentu v integralu
      integralComplexExponent2.im := -integralExponent2; // imaginarni slozka
exponentu v integralu
      uSampleComplex2.im:=0; // imaginarni slozka vzorku akci veliciny
      uSampleComplex2.:=
(uSamplingArray[samplingArrayColumnIndex,indexIntegral])+(uSamplingArray[sampl
ingArrayColumnIndex,indexShiftIntegral]); // realna slozka vzorku akci veliciny
      ySampleComplex2.im:=0; // imaginarni slozka vzorku regulovane veliciny
      ySampleComplex2.re:=
(ySamplingArray[samplingArrayColumnIndex,indexIntegral])+(ySamplingArray[sampl
ingArrayColumnIndex,indexShiftIntegral]); // realna slozka vzorku regulovane
veliciny
      integralU1 := CADD(X := integralU1, Y :=(CMUL(X:=uSampleComplex1, Y:=
CEXP(X :=integralComplexExponent1)))); // jmenovatel zlomku pro vypocet druheho
bodu
      integralY1 := CADD(X := integralY1, Y :=(CMUL(X:=ySampleComplex1, Y:=
CEXP(X :=integralComplexExponent1)))); // citatel zlomku pro vypocet druheho
bodu
      integralU2 := CADD(X := integralU2, Y :=(CMUL(X:=uSampleComplex2, Y:=
CEXP(X :=integralComplexExponent2)))); // jmenovatel zlomku pro vypocet tretiho
bodu
      integralY2 := CADD(X := integralY2, Y :=(CMUL(X:=ySampleComplex2, Y:=
CEXP(X :=integralComplexExponent2)))); // citatel zlomku pro vypocet tretiho
bodu
      indexIntegral := indexIntegral + 1; // navyseni indexu v integralu
      indexShiftIntegral := indexShiftIntegral +1; // navyseni posunuteho indexu
v integralu
      IF indexIntegral = currentLastIndex THEN //konec intervalu
        runningBlockCase := 100; // prechod do dalsiho CASE
      END_IF;
```

```
////////////////////////////////////////////////////////////////////////////
////////////////////////////// vypocet G1 a G2  ////////////////////////////
////////////////////////////////////////////////////////////////////////////

  100: // Init
    runningBlockCase := 101; // prechod do dalsiho CASE

  101: // Main
    G1 := CDIV(X := integralY1, Y := integralU1); // druhy bod
    G2 := CDIV(X := integralY2, Y := integralU2); // treti bod
    runningBlockCase := 110; // prechod do dalsiho CASE

////////////////////////////////////////////////////////////////////////////
////////////////////////// urceni parametru modelu  ////////////////////////
////////////////////////////////////////////////////////////////////////////


  110: // Init
    runningBlockCase := 111; // prechod do dalsiho CASE

  111: // Main
    fb_parametrySOTD(omega_1 := omega1, omega_2 := omega2, K_s := K, bod_1.re
:= G1.re,  bod_1.im := G1.im, bod_2.re := G2.re,  bod_2.im := G2.im, a_2=>sotdA2,
a_1=>sotdA1, T_d=>sotdTau, K_p=>sotdK); // vypocet parameru modelu
    runningBlockCase := 120; // prechod do dalsiho CASE

////////////////////////////////////////////////////////////////////////////
////////////////////// priprava na dalsi identifikaci //////////////////////
////////////////////////////////////////////////////////////////////////////

  120: // Init
    runningBlockCase := 121; // prechod do dalsiho CASE

  121: // Main
    webOutputString  :='Identifikace  soustavy  byla  dokončena';  //  hlaseni
uzivateli
    runningBlockCase := 00; // prechod do dalsiho CASE
    buttonIsPressed := FALSE; // uvolneni tlacitka

  END_CASE;
END_PROGRAM
```

```
// Urceni parametru modelu ze tri bodu Nyquistovy frekvencni charakteristiky
s pridanym dopravnim zpozdenim


VAR_GLOBAL RETAIN
  ySamplingArray : ARRAY [0..2, 0..1300] OF REAL; // vzorkovani prubehu
regulovane veliciny
  uSamplingArray : ARRAY [0..2, 0..1300] OF REAL; //vzorkovani prubehu akcni
veliciny
  samplingArrayLastIndexes : ARRAY [0..2] OF UINT;  //pocty vzorku
  samplingArrayColumnIndex   : USINT := 2; //index používaného sloupce
  yK1 : REAL; // regulovana veličina v bode blizkem pracovnímu bodu
  yK2 : REAL; // regulovana veličina v bode blizkem pracovnímu bodu
  uK1 : REAL; // akcni veličina v bode blizkem pracovnímu bodu
  uK2 : REAL; // akcni veličina v bode blizkem pracovnímu bodu
  trailingEdgeIndex : ARRAY [0..2] OF UINT;  //index na kterém došlo k přepnutí
relé (u)
  currentLastIndex : UINT; // minuly index
END_VAR

VAR_GLOBAL CONSTANT
  BufferArrayLength : UINT := 1200;
END_VAR
VAR_GLOBAL
//vetrak + prutokomer
(*
  prutokomer AT r0_p3_AI1.ENG : REAL; //vstup do PLC
  ventilator AT r0_p3_AO1.ENG : REAL; //vystup z PLC
  u AT ventilator : REAL; //akcni zasah
  y_puvodni AT prutokomer; //vystup soustavy
*)
//u : REAL :=0;
//y : REAL;
//zarovka + termistor
//(*
  termistor AT r0_p3_AI2.ENG : REAL; //vstup do PLC
  zarovka   AT r0_p3_AO0.ENG : REAL; //vystup z PLC
  u AT zarovka : REAL; //akcni zasah
  y AT termistor; //vystup soustavy
//*)
//levitace
(*
  prutokomer AT r0_p3_AI1.ENG : REAL; //vstup do PLC
  ventilator AT r0_p3_AO1.ENG : REAL; //vystup z PLC
  u AT ventilator : REAL; //akcni zasah
  y_puvodni AT prutokomer; //vystup soustavy
  y:REAL;
*)
END_VAR

PROGRAM prgMain
  VAR_INPUT
  END_VAR
  VAR_OUTPUT
  END_VAR
  VAR
  leadingEdgeCounter : INT;  // pocet nabeznych hran
  runningBlockCase : INT :=0;  // aktualni blok kodu
  tempCounter : INT :=0;  // pocet vzorku po detekci ustaleni (yk1, yk2)
  tempCounter2 : INT :=0;  // citac
  trailingEdgeCounter : INT;  // pocet sestupnych hran
  indexIntegral   : UINT;  // index pri integraci
  indexShiftIntegral : UINT;  // posunuty index pri integraci (posuv = Tperiod/2)
  samplingArrayColumnNeighbour1Index : UINT;   // index sousedního sloupce pole
  samplingArrayColumnNeighbour2Index : UINT;   // index sousedního sloupce pole
  samplingArrayIndex   : UINT;   // index pri vzorkovani
  samplingArrayIndexOld : UINT;  // puvodni index pri vzorkovani z predchoziho
pruchodu
```

```
  buttonIsPressed : BOOL :=1;   // input z buttonu
  tempBool : BOOL := false;  // docasna bool hodnota
  tempBool2 : BOOL := false;  // docasna bool hodnota
  samplingExit : bool := false;   // priznak, zda opustit vzorkovani
  hysteresisHighValue : REAL;  // horni mez hystereze
  hysteresisLowValue : REAL;  // dolni mez hystereze
  integralExponent1 : REAL;  // exponent ve vzorci s integralem (realny)
  integralExponent2 : REAL;  // exponent ve druhem vzorci s integralem (realny)
  K :REAL;  // staticka citlivost
  noiseMax : REAL;  // maximalni sum
  omega1 : REAL;   // uhlova rychlost ve vzorci s integralem
  omega2 : REAL;   // uhlova rychlost ve druhem vzorci s integralem
  samplingDelayMax  :  REAL;  //  maximalni  zpozdeni  pri  vzorkovani  pole  (0;
samplingPeriodTimeSpan)
  samplingDelaySum : REAL;  // suma vsech zpozdeni pri vzorkovani pole
  samplingDelaySumMax : REAL;  // maximalni suma vsech zpozdeni pri samplovani
poli
  samplingPeriodTimeSpan :  REAL;   // vzorkovací perioda integrátoru
  sotdA1 : REAL;   // casova konstanta modelu SOTD
  sotdA2 : REAL;   // druha casova konstanta modelu SOTD
  sotdK : REAL;   // staticka citlivost v modelu SOTD
  sotdTau :REAL;   // dopravni zpozdeni v modelu SOTD
  tauIndexIntegral : REAL;   // cas pri integraci
  tempReal : REAL;  // docasna promenna pro real
  tempReal4 : REAL;   // docasna promenna pro real
  tempSum : REAL;   // pomocna promenna pro vypocet prumeru vystupu
  timeDiff : REAL;    // pomocna promenna pro urcovani rozdilu casu
  trafficDelay : REAL;   // dopravni zpozdeni
  T1 : REAL;   // cas pro rele v horni poloze
  T2 : REAL;   // cas pro rele ve spodni poloze
  Tperiod : REAL;   // delka periody [ms]
  u0 : REAL := 3;    // vstup v pracovnim bode
  uHighValue : REAL;  // hodnota vstupu pri horni poloze rele
  uLowValue : REAL;   // hodnota vstupu pri spodni poloze rele
  uMaxValue : REAL := 10;  // maximalni hodnota vstupu
  uMinValue : REAL := 0;  // minimalni hodnota vstupu
  y0 : REAL;  // vystup v pracovnim bode
  ySamplingArrayDiffSum : REAL; // suma rozdilu mezi namerenymi hodnotami Y v
poslednich trech pruchodech
  ySamplingArraySum : REAl;  // suma namerenych hodnot v poli Y
  G1   :  COMPLEX;  // prvni bod Nyquistovy frekvencni charakteristiky v komplexni
rovine
  G2   :  COMPLEX;  // druhy bod Nyquistovy frekvencni charakteristiky v komplexni
rovine
  integralComplexExponent1 : COMPLEX;  // komplexni exponent v prvnim integralu
  integralComplexExponent2 : COMPLEX;  // komplexni exponent ve druhem integralu
  integralU1 : COMPLEX;  // vysledek celeho integralu ve jmenovateli 1. vzorce
  integralU2 : COMPLEX;  // vysledek celeho integralu ve jmenovateli 2. vzorce
  integralY1 : COMPLEX;  // vysledek celeho integralu v citateli 1. vzorce
  integralY2 : COMPLEX;  // vysledek celeho integralu v citateli 2. vzorce
  uSampleComplex1 : COMPLEX;  // vzorek v komplexni rovine ve jmenovateli 1.
vzorce
  uSampleComplex2 : COMPLEX;  // vzorek v komplexni rovine ve jmenovateli 2.
vzorce
  ySampleComplex1 : COMPLEX;  // vzorek v komplexni rovine v citateli 1. vzorce
  ySampleComplex2 : COMPLEX;  // vzorek v komplexni rovine v citateli 2. vzorce
  leadingEdgeRtc : DATE_AND_TIME;  // cas nabezne hrany
  trailingEdgeRtc : DATE_AND_TIME;  // cas sestupne hrany
  tempDateTime : DATE_AND_TIME;  // docasna promenna pro datetime
  T0 : DATE_AND_TIME;  // pocatek vzorkovaci periody
  timer : TON;  // casovac
  timerForDelay : TON;  //casovac pro zpozdeni nabezne a spadove hrany
  webOutputString : STRING;  // vystup na web
  fb_webInputValidation  :  webInputValidation;  //funkci  blok  pro  kontrolu
zadanych udaju
  fb_stabilization : fbstabilization;  //funkcni blok pro kontrolu ustaleni
  fb_parametrySOTD : fbparametrySOTD;  //funkcni blok vypoctu parametru modelu
  END_VAR
```

```
  VAR_TEMP
  END_VAR

  CASE runningBlockCase OF

////////////////////////////////////////////////////////////////////////////
/////////////// nastaveni parametru a cekani na buttonIsDisabled //////////////
////////////////////////////////////////////////////////////////////////////

  00: // Init
    runningBlockCase := 01; // prechod do dalsiho CASE

  01: // Main
    IF buttonIsPressed = TRUE THEN // kontrola stisknuti tlacitka
      fb_webInputValidation(minValue:= uMinValue, minOffset := -0.5, maxValue:=
uMaxValue,  value:=u0,  isValid  =>  tempBool,  message  =>  webOutputString);
//volání podprogramnu pro kontrolu zadaných parametru
      IF tempBool = FALSE THEN // kdyz jsou spatne zadane udaje
        buttonIsPressed := FALSE; //uvolněni tlacitka
      ELSE
        runningBlockCase := 10; // prechod do dalsiho CASE
      END_IF;
    END_IF;

////////////////////////////////////////////////////////////////////////////
//////////////////////////////  cekani na ustaleni  ////////////////////////////
////////////////////////////////////////////////////////////////////////////

  10: // Init
    u:=u0-0.5;    // nastavení vstupního napití pod pracovní bod, kvuli ureení
statické citlivosti
    runningBlockCase := 11; // prechod do dalsiho CASE

  11: // Main
    fb_stabilization(vstup:=y,  cas_mereni:=  T#10s,  ustaleno  =>  tempBool);  //
cekani na ustaleni vystupu
    IF tempBool = TRUE THEN // kdyz je vystup ustaleny
      runningBlockCase := 20; // prechod do dalsiho CASE
    END_IF;

////////////////////////////////////////////////////////////////////////////
//////////////////////////////  urceni u1 a y1  ////////////////////////////////
////////////////////////////////////////////////////////////////////////////

  20: // Init
    tempCounter := 0; //vynulovani citace
    tempSum := 0; //vynulovani promenne
    runningBlockCase := 21; // prechod do dalsiho CASE

  21: // Main
    timer(IN:=TRUE, PT:=T#1s); // casovac pro urceni prumerne hodnoty vystupu v
bode 1 pro urceni staticke citlivosti
    tempSum := tempSum+y;  // suma vystupu po dobu behu casovace
    tempCounter := tempCounter+1; // pocet sectenych vystupu po dobu casovace
    IF timer.Q THEN // po uplynutí easovaee
      timer(IN:=FALSE); // vynulování casovace
      yK1  :=  tempSum/INT_TO_REAL(tempCounter);  //  vypocet  prumerne  hodnoty
vystupu
      uK1 := u0 - 0.5; // hodnota akcni veliciny
      runningBlockCase := 30; // prechod do dalsiho CASE
    END_IF;

////////////////////////////////////////////////////////////////////////////
//////////////////////////// cekani na ustaleni  ////////////////////////////
////////////////////////////////////////////////////////////////////////////

  30: // Init
    u:=u0; //nastavení akcni veliciny na zadanou hodnotu, tj. pracovni bod
```

```
    runningBlockCase := 31; // prechod do dalsiho CASE

  31: // Main
    fb_stabilization(vstup:=y, cas_mereni:= T#10s, ustaleno => tempBool); //
cekani na ustaleni vystupu
    IF tempBool = TRUE THEN // kdyz je ustaleny vystup
       runningBlockCase := 40; // prechod do dalsi faze programu
    END_IF;

////////////////////////////////////////////////////////////////////////////
////////////////////////////// urceni u2 a y2  ////////////////////////////////
////////////////////////////////////////////////////////////////////////////

  40: // Init

    tempSum := 0; // vynulovani promenne
    tempCounter := 0; // vynulovyni citace
    runningBlockCase := 41; // prechod do dalsiho CASE

  41: // Main
    timer(IN:=TRUE, PT:=T#1s); // casovac pro vypocteni prumerne hodnoty vystupu
    tempSum := tempSum+y; // suma vzorku vystupu
    tempCounter := tempCounter+1; //pocet sectenych vystupu
    IF timer.Q THEN // když casovac dobehne
      timer(IN:=FALSE); // nulovani casovace
      yK2 := tempSum/INT_TO_REAL(tempCounter); // vypocet prumerneho vystupu v
bode pro urceni staticke citlivosti
      uK2 := u0; //akcni velicina v bode pro urceni staticke citlivosti
      y0:=yK2; //regulovana velicina v pracovnim bode
      K:=(yK2-yK1)/(uK2-uK1); //vypocet hodnoty statické citlivosti
      runningBlockCase:=50; // prechod do dalsiho CASE
    END_IF;

////////////////////////////////////////////////////////////////////////////
////////////////////////////// mereni sumu ////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

  50: // Init
    noiseMax := 0; //vynulování nejvyšší hodnoty výstupu
    runningBlockCase:=51; // prechod do dalsiho CASE

  51: // Main
    timer(IN:=TRUE, PT:=T#10s); // casovac po ktery se meri sum
    IF timer.Q = FALSE THEN // kdyz casovac nedobehl
      IF ABS(y0-y)>noiseMax THEN // kdyz absolutní hodnota sumu je vetsi nez
nejvetsi ulozena
        noiseMax:=ABS(y0-y); // ulozeni velikosti sumu z aktualni absoulutni
hodnoty sumu
      END_IF;
    ELSE
      timer(IN:=FALSE); // vynulovani casovace
      hysteresisHighValue:=noiseMax*8; // urceni mezi hystereze rele z velikosti
sumu
      hysteresisLowValue:=noiseMax*10; // urceni mezi hystereze rele z velikosti
sumu
      runningBlockCase:=70; // prechod do dalsiho CASE
    END_IF;

  (* // uprava pro pouziti simulace soustavy
    noiseMax:=0.2;
    hysteresisHighValue:=noiseMax*6;
    hysteresisLowValue:=noiseMax*10;
    runningBlockCase :=70;
  *)

////////////////////////////////////////////////////////////////////////////
//////////////////////////////  urceni periody  ////////////////////////////////
////////////////////////////////////////////////////////////////////////////
```

```
  70: // Init
    uHighValue := u0 + 5; // urceni horni polohy rele
    IF uHighValue > uMaxValue THEN // pokud horni poloha rele je vyssi nez
maximalni
      uHighValue:= uMaxValue; // uprava horni hodnoty rele
    END_IF; // horni mez rele

    uLowValue:= u0 - 3; // dolní poloha rele
    IF uLowValue < uMinValue THEN // kdyz je spodni poloha relé mensi nez
minimalni
      uLowValue:= uMinValue; // uprava spodni polohy rele
    END_IF;
    u:= uHighValue; // nastaveni akcni veliciny na horni mez rele
    trailingEdgeCounter:=0; // citac spadovych hran
    leadingEdgeCounter:=1; // citac nabeznych hran
    tempBool := FALSE; // ukazatel stejne delky casu T1
    tempBool2 := FALSE; // ukazatel stejne delky casu T2
    T1 := 1; // nastaveni casu T1 na nenulovou hodnotu
    T2 := 1; // nastaveni casu T2 na nenulovou hodnotu
    leadingEdgeRtc := getRTC(); // cas nabezne hrany
    trailingEdgeRtc := getRTC(); // cas spadove hrany
    runningBlockCase := 71; // prechod do dalsiho CASE

  71: // Main
  IF y>y0+hysteresisHighValue AND u=uHighValue THEN // kdyz je cas na sestupnou
hranu a jeste k ni nedoslo
    u:= uLowValue; // dolni mez rale
    tempDateTime:= getRTC(); // ulozeni aktualniho casu
    trailingEdgeCounter := trailingEdgeCounter + 1; // citac spadovych hran
    trailingEdgeRtc := tempDateTime; // cas spadove hrany
    tempReal  :=  (TIME_TO_REAL(SUB_DT_DT(tempDateTime,leadingEdgeRtc)));  //
aktualni cas T1
    IF abs(1.0 - tempReal/T1) < 0.03 THEN // kdyz je rozdil mezi soucasnym a
minulym T1 rozdil mensi nez 3 procenta
      tempBool := TRUE; // ukazatel splneni podminky
    ELSE
      tempBool := FALSE; // ukazatel na nesplneni podminky
    END_IF;
    T1 := tempReal; // prepsani casu T1
  END_IF;
  IF y<y0-hysteresisLowValue AND u=uLowValue THEN // kdyz ma byt nabezna hrana
a jeste k ni nedoslo
    u:= uHighValue; // nastavi se nabezna hrana
    tempDateTime:= getRTC(); // cas nabezne hrany
    leadingEdgeCounter := leadingEdgeCounter + 1; // citac nabeznych hran
    leadingEdgeRtc := tempDateTime; // cas nabezne hrany
    tempReal  :=  (TIME_TO_REAL(SUB_DT_DT(tempDateTime,trailingEdgeRtc)));  //
vypocet casu T2
    IF abs(1.0 - tempReal/T2) < 0.03 THEN // kdyz je rozdil mezi soucasnym a
minulym T2 rozdil mensi nez 3 procenta
      tempBool2 := TRUE; // ukazatel splneni podminky
    ELSE
      tempBool2 := FALSE; // ukazatel nesplneni podminky
    END_IF;
    T2 := tempReal; // cas T2
  END_IF;
  IF tempBool AND tempBool2 AND u=uHighValue THEN // kdyz jsou splneny podminky
a rele je nahore
    Tperiod := T1+T2; // vypocet periody
    TrafficDelay := Tperiod/2;
    runningBlockCase :=80; // prechod do dalsiho CASE
  END_IF;

////////////////////////////////////////////////////////////////////////////
//////////////////////////  vzorkovani  ////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

  80: // Init
```

```
    IF T1 > T2 THEN // kontrola splneni podminky
      webOutputString := 'T1 je vetsi nez T2, sprav si to';   // hlaseni o
nesplneni podminky
      runningBlockCase := 1000; // prechod do dalsiho CASE
    END_IF;
    samplingArrayIndexOld := 1000; // nastaveni indexu na nesmyslny
    samplingArrayColumnIndex := 0; // nastaveni sloupce pole
    samplingArrayColumnNeighbour1Index := 1; // nastaveni sousedniho sloupce
    samplingArrayColumnNeighbour2Index := 2; // nastaveni druheho sousedniho
sloupce
    ySamplingArrayDiffSum := 0; // nulovani souctu rozdilu mezi sloupci
    ySamplingArraySum := 0; // soucet ulozenych prvku
    samplingArrayLastIndexes[0] := 1000; // nastaveni indexu na nesmyslny
    samplingArrayLastIndexes[1] := 1000; // nastaveni indexu na nesmyslny
    samplingArrayLastIndexes[2] := 1000; // nastaveni indexu na nesmyslny
    tempCounter := 0; // citac
    tempCounter2 := 0; // citac
    samplingDelayMax := 0; // nulovani maximalniho zpozdeni
    samplingDelaySum := 0; // nulovani souctu zpozdeni
    samplingDelaySumMax := 0; // nulovani maximalniho celkoveho souctu zpozdeni
    samplingPeriodTimeSpan := Tperiod / 900.0; // vzorkovaci perioda [ms], ale
opravdu! [ms]!!
    samplingPeriodTimeSpan  :=  10*CEIL(samplingPeriodTimeSpan/10);  //  uprava
vzorkovaci periody
    IF (samplingPeriodTimeSpan < 20) THEN  // kdyz je vzorkovaci perioda mensi
nez 20ms
      samplingPeriodTimeSpan := 20; // vzorkovaci perioda 20ms
    END_IF;
    T0:=GetRTC(); // aktualni cas
    runningBlockCase := 81; // prechod do dalsiho CASE

  81: // Main
    IF y>y0+hysteresisHighValue AND u=uHighValue THEN // kdyz je cas na sestupnou
hranu
      timerForDelay(IN := TRUE, PT := REAL_TO_TIME(trafficDelay)); // zpozdeni
sestupne hrany
      IF timerForDelay.Q = TRUE THEN  // kdyz casovac dobehne
        u:= uLowValue; // sestupna hrana
        tempCounter2 := tempCounter2+1; //citac
        timerForDelay(IN := FALSE); // vynulovani casovace
        tempDateTime:= getRTC();  // aktualni cas
        trailingEdgeRtc := tempDateTime; // cas spadove hrany
        tempReal := (TIME_TO_REAL(SUB_DT_DT(tempDateTime,leadingEdgeRtc))); //
T1
        T1 := tempReal; // T1
      END_IF;
    END_IF;
    IF y<y0-hysteresisLowValue AND u=uLowValue THEN // cas na nabeznou hranu
      timerForDelay(IN := TRUE, PT := REAL_TO_TIME(trafficDelay)); // zpozdeni
nabezne hrany
      IF timerForDelay.Q = TRUE THEN // dobehl casovac
        u:= uHighValue; // nabezna hrana
        timerForDelay(IN := FALSE); // vynulovani casovace
        tempDateTime:= getRTC(); // aktualni cas
        leadingEdgeRtc := tempDateTime; //cas nabezne hrany
        tempReal := (TIME_TO_REAL(SUB_DT_DT(tempDateTime,trailingEdgeRtc))); //
T2
        T2 := tempReal; // T2
        T0:= getRTC(); // aktualni cas
        IF (samplingDelaySum > samplingDelaySumMax) THEN // kontrola hodnoty
sumy zpozdeni
          samplingDelaySumMax := samplingDelaySum; // prepsani hodnoty sumy
zpozdeni
        END_IF;
        samplingDelaySum := 0; // vynulovani sumy zpozdeni
        IF tempCounter2 > 3 AND ySamplingArrayDiffSum/ySamplingArraySum < 0.06
THEN  // kontrola opousteci podminky
          samplingExit := TRUE; // ukazatel ukonceni vzorovani
```

```
          runningBlockCase := 90; // prechod do dalsiho CASE
        ELSE
          samplingArrayColumnIndex := samplingArrayColumnIndex + 1; // prechod
do dalsiho sloupce
        IF samplingArrayColumnIndex > 2 THEN  // kontrola cisla sloupce
          samplingArrayColumnIndex := 0; // uprava cisla sloupce
        END_IF;
        samplingArrayLastIndexes[samplingArrayColumnIndex] := 0; // vynulovani
cisla posledniho vzorku
        CASE samplingArrayColumnIndex OF  // upravy sloupcu sousedu
        0 :
          samplingArrayColumnNeighbour1Index := 1;
          samplingArrayColumnNeighbour2Index := 2;
        1 :
          samplingArrayColumnNeighbour1Index := 0;
          samplingArrayColumnNeighbour2Index := 2;
        2:
          samplingArrayColumnNeighbour1Index := 0;
          samplingArrayColumnNeighbour2Index := 1;
        END_CASE;
        ySamplingArraySum := 0; // vynulovani souctu vzorku
        ySamplingArrayDiffSum := 0; // vynulovani souctu rozdilu vzorku
      END_IF;
    END_IF;
    END_IF;
    IF samplingExit = FALSE THEN  // kdyz neni ukonceno vzorkovani
      timeDiff  :=  TIME_TO_REAL(SUB_DT_DT(GetRTC(),T0));  //  cas  od  pocatku
periody
      samplingArrayIndex:=            REAL_TO_UINT(FLOOR(timeDiff            /
samplingPeriodTimeSpan)); // index vzorku
      IF (samplingArrayIndex <> samplingArrayIndexOld) THEN // kdyz mame novy
index vzorku
        ySamplingArray[samplingArrayColumnIndex,samplingArrayIndex]  :=  y;  //
ulozeni vzorku regulovane veliciny
        uSamplingArray[samplingArrayColumnIndex,samplingArrayIndex]  :=  u;  //
ulozeni vzorku akcni veliciny
        samplingArrayLastIndexes[samplingArrayColumnIndex]:=
samplingArrayIndex; // prepsani posledniho indexu
        IF   samplingArrayLastIndexes[samplingArrayColumnNeighbour1Index]   >=
samplingArrayIndex THEN // pokud nebyl presazen pocet vzorku v sousednim sloupci
          ySamplingArrayDiffSum    :=    ySamplingArrayDiffSum    +    abs(y-
ySamplingArray[samplingArrayColumnNeighbour1Index   ,samplingArrayIndex]);   //
prepocet sumy rozdilu
        END_IF;
        IF   samplingArrayLastIndexes[samplingArrayColumnNeighbour2Index]   >=
samplingArrayIndex THEN // pokud nebyl presazen pocet vzorku v sousednim sloupci
          ySamplingArrayDiffSum    :=    ySamplingArrayDiffSum    +    abs(y-
ySamplingArray[samplingArrayColumnNeighbour2Index   ,samplingArrayIndex]);   //
prepocet sumy rozdilu
        END_IF;
        ySamplingArraySum := ySamplingArraySum + abs(y); // suma ulozenych prvku
        IF samplingArrayIndex = 0 THEN // kdyz je pocatek periody
          tempReal4 := timeDiff; // rozdil casu
        ELSE
          tempReal4  :=  timeDiff  -  (UINT_TO_REAL(samplingArrayIndexOld  +  1)  *
samplingPeriodTimeSpan); // rozdil casu
        END_IF;
        IF (tempReal4 > samplingDelayMax) THEN  // kontrola maximalniho zpozdeni
          samplingDelayMax := tempReal4; // maximalni zpozdeni
        END_IF;
        samplingDelaySum  :=  samplingDelaySum  +  tempReal4;  //  suma  zpozdeni
vzorkovani
        IF    (samplingArrayIndex    <>    0    AND    samplingArrayIndex    <>
samplingArrayIndexOld + 1) THEN
          tempCounter := tempCounter + 1; // citac vzorku
        END_IF;
        samplingArrayIndexOld := samplingArrayIndex; // minuly index
      END_IF;
```

```
    END_IF;

////////////////////////////////////////////////////////////////////////////
////////////////////////// vypocet integralu ///////////////////////////////
////////////////////////////////////////////////////////////////////////////

  90: // Init
    Tperiod := T1+ T2; // vypocet periody
    omega1   := MATH.PI*2 / (Tperiod * 0.001); // frekvence identifikace
    omega2  := 2.0*omega1; // dvojnasobek frekvence identifikace
    indexIntegral:=0; // index prvku pro integraci
    indexShiftIntegral:=
(samplingArrayLastIndexes[samplingArrayColumnIndex]/2); // posunuty index prvku
pro integraci
    currentLastIndex:=  samplingArrayLastIndexes[samplingArrayColumnIndex];  //
posledni index vzorku
    u :=uMinValue; // minimalizovani hodnoty akcni veliciny
    runningBlockCase := 91; // prechod do dalsiho CASE

  91: // Main
      IF  indexShiftIntegral  >  currentLastIndex  THEN  //  kontrola  posunuteho
indexu v integralu
        indexShiftIntegral := indexShiftIntegral - currentLastIndex; // uprava
posunuteho indexu v integralu
      END_IF;
      tauIndexIntegral    := UINT_TO_REAL(indexIntegral)*samplingPeriodTimeSpan
/ 1000; // cas v integralu jako desetiné císlo v sekundách
      integralExponent1:= omega1*tauIndexIntegral; // exponent integralu
      integralExponent2:= omega2*tauIndexIntegral; // exponent integralu
      integralComplexExponent1.re := 0; // realna slozka exponentu v integralu
      integralComplexExponent1.im := -integralExponent1; // imaginarni slozka
exponentu v integralu
      uSampleComplex1.im:=0; // imaginarni slozka vzorku akci veliciny
      uSampleComplex1.re:=
(uSamplingArray[samplingArrayColumnIndex,indexIntegral]);   //   realna   slozka
vzorku akci veliciny
      ySampleComplex1.im:=0; // imaginarni slozka vzorku regulovane veliciny
      ySampleComplex1.re:=
(ySamplingArray[samplingArrayColumnIndex,indexIntegral]);   //   realna   slozka
vzorku regulovane veliciny
      integralComplexExponent2.re := 0; // relna slozka exponentu v integralu
      integralComplexExponent2.im := -integralExponent2; // imaginarni slozka
exponentu v integralu
      uSampleComplex2.im:=0; // imaginarni slozka vzorku akci veliciny
      uSampleComplex2.re:=
(uSamplingArray[samplingArrayColumnIndex,indexIntegral])+(uSamplingArray[sampl
ingArrayColumnIndex,indexShiftIntegral]); // realna slozka vzorku akci veliciny
      ySampleComplex2.im:=0; // imaginarni slozka vzorku regulovane veliciny
      ySampleComplex2.re:=
(ySamplingArray[samplingArrayColumnIndex,indexIntegral])+(ySamplingArray[sampl
ingArrayColumnIndex,indexShiftIntegral]); // realna slozka vzorku regulovane
veliciny
      integralU1 := CADD(X := integralU1, Y :=(CMUL(X:=uSampleComplex1, Y:=
CEXP(X :=integralComplexExponent1)))); // jmenovatel zlomku pro vypocet druheho
bodu
      integralY1 := CADD(X := integralY1, Y :=(CMUL(X:=ySampleComplex1, Y:=
CEXP(X :=integralComplexExponent1)))); // citatel zlomku pro vypocet druheho
bodu
      integralU2 := CADD(X := integralU2, Y :=(CMUL(X:=uSampleComplex2, Y:=
CEXP(X :=integralComplexExponent2)))); // jmenovatel zlomku pro vypocet tretiho
bodu
      integralY2 := CADD(X := integralY2, Y :=(CMUL(X:=ySampleComplex2, Y:=
CEXP(X :=integralComplexExponent2)))); // citatel zlomku pro vypocet tretiho
bodu
      indexIntegral := indexIntegral + 1; // navyseni indexu v integralu
      indexShiftIntegral := indexShiftIntegral +1; // navyseni posunuteho indexu
v integralu
      IF indexIntegral = currentLastIndex THEN //konec intervalu
```

```
        runningBlockCase := 100; // prechod do dalsiho CASE
      END_IF;

////////////////////////////////////////////////////////////////////////////
///////////////////////////// vypocet G1 a G2  /////////////////////////////
////////////////////////////////////////////////////////////////////////////


  100: // Init
    runningBlockCase := 101; // prechod do dalsiho CASE

  101: // Main
    G1 := CDIV(X := integralY1, Y := integralU1); // druhy bod
    G2 := CDIV(X := integralY2, Y := integralU2); // treti bod
    runningBlockCase := 110; // prechod do dalsiho CASE

////////////////////////////////////////////////////////////////////////////
///////////////////////////// urceni parametru modelu  /////////////////////
////////////////////////////////////////////////////////////////////////////


  110: // Init
    runningBlockCase := 111; // prechod do dalsiho CASE

  111: // Main
    fb_parametrySOTD(omega_1 := omega1, omega_2 := omega2, K_s := K, bod_1.re
:= G1.re,  bod_1.im := G1.im, bod_2.re := G2.re,  bod_2.im := G2.im, a_2=>sotdA2,
a_1=>sotdA1, T_d=>sotdTau, K_p=>sotdK); // vypocet parameru modelu
    runningBlockCase := 120; // prechod do dalsiho CASE

////////////////////////////////////////////////////////////////////////////
///////////////////// priprava na dalsi identifikaci ///////////////////////
////////////////////////////////////////////////////////////////////////////

  120: // Init
    runningBlockCase := 121; // prechod do dalsiho CASE

  121: // Main
    webOutputString  :='Identifikace  soustavy  byla  dokončena';  //  hlaseni
uzivateli
    runningBlockCase := 00; // prechod do dalsiho CASE
    buttonIsPressed := FALSE; // uvolneni tlacitka

  END_CASE;
END_PROGRAM
```

```
// Urceni modelu pro nastaveni parametru PI, PD a PID regulatoru

VAR_GLOBAL
//vetrak + prutokomer
(*
prutokomer AT r0_p3_AI1.ENG : REAL; //vstup do PLC
  ventilator AT r0_p3_AO1.ENG : REAL; //výstup z PLC
   u AT ventilator : REAL; //akční zásah
  y AT prutokomer; //výstup soustavy
*)
//u: REAL;
//y: REAL;

//zarovka + termistor
termistor AT r0_p3_AI2.ENG : REAL; //vstup do PLC
zarovka   AT r0_p3_AO0.ENG : REAL; //výstup z PLC
 u AT zarovka : REAL; //akční zásah
 y AT termistor; //výstup soustavy

END_VAR

PROGRAM prgMain
  VAR_INPUT
  END_VAR
  VAR_OUTPUT
  END_VAR
  VAR

  buttonIsPressed : BOOL:=TRUE;  // startovaci tlacitko
  tempBool : BOOL;  // docana promenna typu BOOL
  tempCounter : INT;  // pomocny citac
  leadingEdgeCounter : UINT :=0;  // citac nabeznych hran
  runningBlockCase : UINT :=00;  // ukazatel na CASE
  samplingArrayColumnIndex :UINT := 2;  // index pouzivaneho bufferu
  samplingArrayIndex :UINT;  // index v bufferu
  samplingArrayIndexOld : UINT;  // index predchoziho vzorku v bufferu
  trailingEdgeCounter : UINT :=0; // citac spadovych hran
  yConditions : INT := 0; // ukazatel na CASE
  coefRelay : REAL; // koeficient pro přepočet poloh relé
  deviation : REAL := 0.1;  // povolené rozdíly naměřených průběhů při 1 periodě
  divisionTdT :REAL; // pomer (tau)/(1-tau)
  exponentialCorrectionCoeficient : REAL := 1; // koeficient pro úpravu doby
náběhu exponenciely
  gama : REAL; // stupen asymetrie rele
  hysteresisHighValue : REAL; // horní odchylka od žádané hodnoty y
  hysteresisLowValue : REAL; // dolni odchylka od žádané hodnoty y
  K : REAL; // staticka citlivost
  Td : REAL; // dopravní zpoždění
  noiseMax : REAL; // maximalni sum
  ny : REAL := 5.0; // koeficient pro vypocet yMaxLowDeviation
  ro : REAL; // pomer casu, po ktery je rele v horni respektive dolni poloze
  roTemp : REAL; // pomer casu, po ktery je rele v horni respektive dolni poloze
- pomocný
  samplingPeriodTimeSpan : REAL :=0.01; // vzorkovací perioda numerické
integrace
  T : REAL; // časová konstanta
  tau : REAL; // nomralizovane dopravni zpozdeni
  tauTemp : REAL; // nomralizovane dopravni zpozdeni - pomocné
  tempReal : REAL := 0; // docasna promenna typu REAL
  tempReal2 : REAL := 0; // docasna promenna typu REAL
  tempSum : REAL; // docasna Suma regulovane veliciny
  u0 : REAL:=3; // akcni velicina v pracovnim bode
  uHighValue : REAL; // zadaná velikost akčního zásahu při horní poloze relé
  uLowValue : REAL; // zadaná velikost akčního zásahu při dolní poloze relé
  uK1 : REAL; // akcni velicina v bode blizkem pracovniho bodu
  uK2 : REAL; // akcni velicina v pracovnim bode
  uMaxValue : REAL :=6.0; // maximální dovolená hodnota vstupu
  uMinValue : REAL := 1.5; // minimální dovolená hodnota vstupu
```

```
  uValueTemp : REAL; // pomocna hodnota akcni veliciny
  uValueTemp2 :REAL; // pomocna hodnota akcni veliciny
  y0 : REAL; // regulovana velicina v pracovnim bode
  yDeflectionHigh : REAL := 0.0; // maximalni vychylka nad zadanou hodnotou
  yDeflectionHighLast : REAL; // predchozi hodnota yDeflectionHigh
  yDeflectionLow : REAL := 0.0; // maximalni vychylka pod zadanou hodnotou
  yDeflectionLowLast : REAL; // predchozi hodnota yDeflectionLow
  yK1 : REAL; // regulovana velicina v bode blizkem pracovnimu bodu
  yK2 : REAL; // regulovana velicina v pracovnim bode
  yMaxHighDeviation : REAL;   // horni hranice pasma pro vystup soustavy pri
zapojeni rele
  yMaxLowDeviation : REAL;   // dolni hranice pasma pro vystup soustavy pri
zapojeni rele
  ySamplingArrayDiff01 : REAL; // prumerne rozdily mezi buffery 0 a 1
  ySamplingArrayDiff02 : REAL; // prumerne rozdily mezi buffery 0 a 2
  ySamplingArrayDiff12 : REAL; // prumerne rozdily mezi buffery 1 a 2
  ySamplingArrayDiffA : REAL;  // suma rozdilu
  ySamplingArrayDiffB : REAL;  // suma rozdilu
  exponentialTimeSpan : TIME; // exponent
  T1 : TIME; // doba horni polohy rele
  T1Temp : TIME; // doba horni polohy rele - pomocna
  T2 : TIME; // doba dolni polohy rele
  T2Temp : TIME; // doba dolni polohy rele - pomocna
  timeInPeriod: TIME; // cas od pocatku periody
  Tperiod :TIME; // doba trvani jedné periody
  currentTime : DT; // aktualni cas
  leadingEdgeRtc : DT; // casová značka prepnuti na uHighValue
  trailingEdgeRtc : DT; // casová značka prepnuti na uLowValue
  leadingEdgeRtcTemp : DT; // casová značka prepnuti na uHighValue - pomocna
  startExonentialTime : DT; // cas začatku rustu exponenciely
  trailingEdgeRtcTemp : DT; // casova znacka prepnuti na uLowValue - pomocna
  timer : TON;  // casovac
  fb_stabilization :  fbstabilization ; // funkcni blok ustaleni
  fb_webInputValidation : fbwebInputControl; // funkcni blok kontroly zadanych
parametru
  samplingArrayLastIndexes : ARRAY [0..2] OF UINT;  // pocet ulozenych vzorku
  trailingEdgeIndex : ARRAY [0..2] OF UINT; // index na kterem doslo k prepnuti
rele
  ySamplingArray : ARRAY [0..2,0..2999] OF REAL; // pole pro ukladani vzorku
regulovane veliciny
  webOutputString : STRING; // textove pole pro vypis informaci pro uzivatele

  END_VAR
  VAR_TEMP
  END_VAR

  CASE runningBlockCase OF

////////////////////////////////////////////////////////////////////////////
/////////////// nastaveni parametru a cekani na buttonIsDisabled ///////////
////////////////////////////////////////////////////////////////////////////

  00: // Init
    runningBlockCase := 01; // prechod do dalsiho CASE

  01: // Main
    IF buttonIsPressed = TRUE THEN // kdyz je tlacitko stisknuto
       fb_webInputValidation(minValue:=   uMinValue,   minOffset   :=   -0.5,
maxValue:=   uMaxValue,   value:=u0,   isValid   =>   tempBool,   message   =>
webOutputString);  //volání podprogramnu pro kontrolu zadaných parametru
       IF tempBool = false then // kdyz nejsou spravne zadane parametry
          buttonIsPressed := false; // tacitko neni stisknute
       ELSE
           runningBlockCase := 10; // prechod do dalsiho CASE
       END_IF;
    END_IF;
```

```
////////////////////////////////////////////////////////////////////////////
////////////////////////////// cekani na ustaleny //////////////////////////
////////////////////////////////////////////////////////////////////////////

  10: // Init
    u:=u0-0.5; // nastavení vstupního napití pod pracovní bod, kvuli urceni
staticke citlivosti
    runningBlockCase := 11; // prechod do dalsiho CASE

  11: // Main
    fb_stabilization(vstup:=y, cas_mereni:= T#10s, ustaleno => tempBool); //
cekani na ustaleni vystupu
    IF tempBool = TRUE THEN // kdyz je vystup ustaleny
      runningBlockCase := 20; //prechod do dalsiho CASE
    END_IF;

////////////////////////////////////////////////////////////////////////////
////////////////////////////// urceni u1 a y1  /////////////////////////////////
////////////////////////////////////////////////////////////////////////////

  20: // Init
    tempCounter := 0; // vynulovani citace
    tempSum := 0; // vynulovani sumy regulovane veliciny
    runningBlockCase := 21; // prechod do dalsiho CASE

  21: // Main
    timer(IN:=TRUE, PT:=T#1s); // casovac pro urceni prumerne hodnoty vystupu v
bode 1 pro urceni staticke citlivosti
    tempSum := tempSum+y;  // sumy vystupu po dobu behu casovace
    tempCounter := tempCounter+1; //citac pruchodu programem po dobu casovace
    IF timer.Q THEN // po uplynutí casovace
      timer(IN:=FALSE); // vynulovani casovace
      yK1  :=  tempSum/INT_TO_REAL(tempCounter);  //  vypocet  prumerne  hodnoty
vystupu
      uK1 := u0 - 0.5; // hodnota akení velieiny
      runningBlockCase := 30; //predem do dalsiho CASE
    END_IF;

////////////////////////////////////////////////////////////////////////////
////////////////////////////// cekani na ustaleni //////////////////////////
////////////////////////////////////////////////////////////////////////////

  30: // Init
    u:=u0; // nastaveni akcni veliciny na zadanou hodnotu, tj. pracovní bod
    runningBlockCase := 31; // prechod do dalsiho CASE

  31: // Main
    fb_stabilization(vstup:=y, cas_mereni:= T#10s, ustaleno => tempBool); //
cekani na ustaleni soustavy
    IF tempBool = TRUE THEN // kdyz je ustaleny vystup
       runningBlockCase := 40; // prechod dod dalsiho CASE
    END_IF;

////////////////////////////////////////////////////////////////////////////
////////////////////////////// urceni u2 a y2  /////////////////////////////////
////////////////////////////////////////////////////////////////////////////

  40: // Init
    tempSum := 0; // vynulovani sumy vzorku regulovane veliciny
    tempCounter := 0; // vynulovani citace
    runningBlockCase := 41; // prechod do dalsiho CASE

  41: // Main
    timer(IN:=TRUE, PT:=T#1s); // casovac pro namereni hodnoty sumu
    tempSum := tempSum+y; // suma vzorku regulovane veliciny
    tempCounter := tempCounter+1; // citac sectenych vzorku
    IF timer.Q THEN // kdyz dobehne casovac
```

```
      timer(IN:=FALSE); // nulovani casovace
      yK2 := tempSum/INT_TO_REAL(tempCounter); // vypocet prumerne hodnoty akcni
veliciny v pracovnim bode
      uK2 := u0; // hodnota akcni veliciny v pracovnim bode
      y0:=yK2; // regulovana velicina v pracovnim bode
      K:=(yK2-yK1)/(uK2-uK1); // staticka citlivost
      runningBlockCase:=50; // prechod do dalsiho CASE
    END_IF;

////////////////////////////////////////////////////////////////////////////
////////////////////////////////// mereni sumu ////////////////////////////////
////////////////////////////////////////////////////////////////////////////

  50: // Init
    noiseMax := 0; // vynulovani nejvyssi hodnoty vystupu
    runningBlockCase:=51; // prechod do dalsiho CASE

  51: // Main
    timer(IN:=TRUE, PT:=T#10s);  // casovac po ktery se meri sum

    IF timer.Q = FALSE THEN // kdyz casovac jeste nedobehl
      IF ABS(y0-y)>noiseMax THEN //kdyz absolutni hodnota sumu je mensi nez
aktualni
        noiseMax:=ABS(y0-y); // prepsani hodnoty sumu
      END_IF;
    END_IF;
    IF timer.Q = TRUE THEN // kdyz casovac dobehl
      timer(IN:=FALSE); // vynulovani casovace
      hysteresisHighValue:=noiseMax*10;  // mez hystereze rele
      hysteresisLowValue:=hysteresisHighValue; // mez hystereze rele
      runningBlockCase:=60; // prechod do dalsiho CASE
    END_IF;

    (*///simulace // prepsani na mereni simulace
    noiseMax:=0.01;
    hysteresisHighValue:=noiseMax*10;
    hysteresisLowValue:=noiseMax*10;
    runningBlockCase:=60;
    ///   *)

    yMaxLowDeviation := ny * hysteresisLowValue; // urceni maximalni horni
hodnoty regulovane veliciny
    yMaxHighDeviation := 3.0*yMaxLowDeviation; // urceni maximalni spodni
hodnoty regulovane veliciny
    uHighValue:= uMaxValue; // horni poloha rele
    uLowValue:= uMinValue; // spodni poloha rele
    gama  :=  MAX(ABS(uHighValue-u0),  ABS(u0-uLowValue))/MIN(ABS(uHighValue-
u0),ABS(u0-uLowValue)); // vypocet gamy

////////////////////////////////////////////////////////////////////////////
////////////////////////// expomencialni prubeh vstupu ////////////////////////
////////////////////////////////////////////////////////////////////////////

  60: // Init
    webOutputString:= 'Nastavuji se hodnoty pro polohy rele'; // hlaseni pro
uzivatele
    startExonentialTime := GetRTC(); // nacteni casu pocatku eponencialy
    runningBlockCase := 61; // prechod do dalsiho CASE

  61: // Main
    currentTime := GetRTC(); //nacteni aktualniho casu
    exponentialTimeSpan := SUB_DT_DT(currentTime,startExonentialTime); // cas
od pocatku exponencialy
    exponentialCorrectionCoeficient:=1/3;   //   koeficient   pro   zpomaleni
exponencialniho rustu
    IF u < uMaxValue THEN //kontrola, ze akcni velicina není vetsi nez povolena
       IF y>=(yMaxHighDeviation+y0) THEN //kontrola, ze vystupni velicina není
vetsi nez zadana velicina s maximalni dovolenou vychylkou
```

```
          runningBlockCase :=70; // prechod do dalsiho CASE
          uValueTemp :=u; // ulozeni aktualni hodnoty akcni veliciny
          uValueTemp2:=u0-((uValueTemp-u0)*0.7);  //  prepocet  hodnoty  akcni
veliciny
          IF uMinValue < uValueTemp2 THEN // kontrola pripravene hodnoty spodni
polohy rele
          uLowValue:= uValueTemp2; // nastaveni spodni polohy rele
          END_IF;
          u:=uLowValue; // nastaveni akcni hodnoty na spodni polohu rele
          uHighValue := uValueTemp; // nastaveni horni hodnoty rele
          gama:= MAX(ABS(uHighValue-u0), ABS(u0-uLowValue))/MIN(ABS(uHighValue-
u0),ABS(u0-uLowValue)); // vypocet gamy
       ELSE
           u:=
EXP(exponentialCorrectionCoeficient*TIME_TO_REAL(exponentialTimeSpan)/1000.0)-
1.0+u0; //exponencielni narust akcni veliciny
       END_IF;
    ELSE
        uValueTemp2:=u0-((uHighValue-u0)*0.7); // vypocet pomocne hodnoty akcni
veliciny
        IF uMinValue < uValueTemp2 THEN // kontrola potencialni hodnoty spodni
polohy rele
          uLowValue:= uValueTemp2; // spodni poloha rele
        END_IF;
       runningBlockCase :=70; // prechod do dalsiho CASE
    END_IF;

////////////////////////////////////////////////////////////////////////////
////////////////////////////// uprava hodnot rele //////////////////////////
////////////////////////////////////////////////////////////////////////////

  70: // Init
       u:=uLowValue; // nastaveni akcni veliciny do spodni polohy rele
       trailingEdgeCounter := 0; // vynulovani citace spadovych hran
       trailingEdgeRtcTemp:=GetRTC(); // cas spadove hrany
       leadingEdgeRtcTemp:=GetRTC(); // cas nabezne hrany
       gama  :=  MAX(ABS(uHighValue-u0),  ABS(u0-uLowValue))/MIN(ABS(uHighValue-
u0),ABS(u0-uLowValue)); // gama
       runningBlockCase :=71; // prechod do dalsiho CASE

  71: // Main
    IF leadingEdgeCounter < 2 THEN // kdyz nebyly dve nabezne hrany
       IF u <> uLowValue THEN // nebyla spadova
         IF y >= y0+hysteresisHighValue THEN // ma byt spadova
            u:= uLowValue; // spadova
           T1Temp  :=  SUB_DT_DT(trailingEdgeRtcTemp,leadingEdgeRtcTemp);  //  T1
docasny
           trailingEdgeRtcTemp  := GetRTC(); // cas spadove
           trailingEdgeCounter := trailingEdgeCounter +1; // citac spadove
         END_IF;
       END_IF;
       IF u <> uHighValue THEN // nebyla nabezna
          IF y <= y0-hysteresisLowValue THEN // mela by byt nabezna
             u:= uHighValue; // nabezna
             T2Temp  :=  SUB_DT_DT(leadingEdgeRtcTemp,trailingEdgeRtcTemp);  //T2
docasny
             leadingEdgeRtcTemp   := GetRTC(); // cas nabezne
             leadingEdgeCounter  :=  leadingEdgeCounter  +1;  //  citac  nabeznych
hran
          END_IF;
       END_IF;
       IF trailingEdgeCounter = 1 AND leadingEdgeCounter = 1 THEN // kdyz je 1
nabezna a 1 spadova
          yDeflectionHigh:= MAX(yDeflectionHigh,ABS(y-y0)); // urceni maximalni
vychylky vystupu nad zadanou velicinu
       END_IF;
       IF trailingEdgeCounter = 0 AND leadingEdgeCounter = 1 THEN // kdyz je 0
spadovych a 1 nabezna
```

```
          yDeflectionLow := MAX(yDeflectionLow,ABS(y0-y)); // urceni maximalni
vychylky vystupu pod zadanou velicinu
       END_IF;
    END_IF;
    IF leadingEdgeCounter = 2  AND trailingEdgeCounter = 1 THEN // 2 nabezne a
1 spadova
       leadingEdgeCounter := 1; // citac nabeznych
       trailingEdgeCounter  := 0; // citac spadovych
       roTemp:=
MAX(TIME_TO_REAL(T1Temp),TIME_TO_REAL(T2Temp))/MIN(TIME_TO_REAL(T1Temp),TIME_T
O_REAL(T2Temp)); // vypocet pomocneho ro
       tauTemp := (gama - roTemp)/((gama-1.0)*((0.35*roTemp)+0.65)); // vypocet
pomocneho tau
       IF tauTemp < 0.05 THEN    // uprava parametru rele
          CASE yConditions OF
          0:
           IF yDeflectionHigh -hysteresisHighValue > 1.0 AND  yDeflectionLow -
hysteresisLowValue > 1.0 THEN // kdyz vychylka na obe starany je vetsi nez 1
//////////////////////////////ZMENA/////////////////////////////////////
             uHighValue:= u0 +(uHighValue - u0)* 0.8; // uprava horni polohy
rele
             uLowValue:= u0 +(uLowValue - u0)* 0.8; // uprava spodni polohy rele
             IF  uHighValue  >  uMaxValue  THEN  //  uprava  horni  polohy  podle
maximalni mozne hodnoty akcni veliciny
                uHighValue:= uMaxValue;
             END_IF;
             IF uLowValue < uMinValue THEN // uprava dolni polohy podle minimalni
mozne hodnoty akcni veliciny
                uLowValue:= uMinValue;
             END_IF;
          END_IF;
          IF yDeflectionHigh -hysteresisHighValue < 0.1 AND  yDeflectionLow -
hysteresisLowValue < 0.1 THEN // kdyz je vychylka mensi nez 0.1
          IF uHighValue = uValueTemp  AND uLowValue = uValueTemp2 THEN // kdyz
se rovnaji pomocnym
             yConditions:= 1; // prechod do CASE
             RETURN;
             ELSE
             uHighValue:= u0 +(uHighValue - u0)* 1.1;  // uprava horni polohy
rele
             uLowValue:= u0 +(uLowValue - u0)* 1.1;  // uprava spodni polohy
rele
             IF  uHighValue  >  uMaxValue  THEN  //  uprava  horni  polohy  podle
maximalni mozne hodnoty akcni veliciny
               uHighValue:= uMaxValue;
             END_IF;
             IF  uLowValue  <  uMinValue  THEN   //  uprava  spodni  polohy  podle
minimalni mozne hodnoty akcni veliciny
                uLowValue:= uMinValue;
             END_IF;
          END_IF;
          ELSE
          IF yDeflectionHigh-hysteresisHighValue <0.1 THEN
             uHighValue:= uHighValue + 0.1; // uprava horni polohy rele
             uLowValue:= uLowValue + 0.1; // uprava spodni polohy rele
             IF  uHighValue  >  uMaxValue  THEN  //  uprava  horni  polohy  podle
maximalni mozne hodnoty akcni veliciny
                uHighValue:= uMaxValue;
             END_IF;
             IF uLowValue < uMinValue THEN // uprava dolni polohy podle minimalni
mozne hodnoty akcni veliciny
                uLowValue:= uMinValue;
             END_IF;
          END_IF;

          IF yDeflectionLow-hysteresisLowValue <0.1 THEN
             uHighValue:= uHighValue - 0.1; // uprava horni polohy rele
             uLowValue:= uLowValue - 0.1; // uprava spodni polohy rele
```

```
            IF uHighValue > uMaxValue THEN // uprava horni polohy podle
maximalni mozne hodnoty akcni veliciny
               uHighValue:= uMaxValue;
            END_IF;
            IF uLowValue < uMinValue THEN // uprava spodni polohy podle
minimalni mozne hodnoty akcni veliciny
               uLowValue:= uMinValue;
            END_IF;
         END_IF;
      END_IF;

//////////////////////////////////////ZMENA////////////////////////////////
       IF yDeflectionHigh -hysteresisHighValue < 1.0 AND  yDeflectionLow -
hysteresisLowValue < 1.0 AND  yDeflectionHigh -hysteresisHighValue > 0.5 AND
yDeflectionLow -hysteresisLowValue > 0.5 THEN
          uHighValue:= u0 +(uHighValue - u0)* 0.9; // uprava horni polohy rele
          uLowValue:= u0 +(uLowValue - u0)* 0.9; // uprava spodni polohy rele
       END_IF;
       IF yDeflectionHigh -hysteresisHighValue < 0.5 AND  yDeflectionLow -
hysteresisLowValue < 0.5 AND  yDeflectionHigh -hysteresisHighValue > 0.1 AND
yDeflectionLow -hysteresisLowValue > 0.1 THEN
          uHighValue:= u0 +(uHighValue - u0)* 0.95;  // uprava horni polohy
rele
          uLowValue:= u0 +(uLowValue - u0)* 0.95; // uprava spodni polohy rele
       END_IF;

//////////////////////////////////////ZMENA////////////////////////////////
     1:
      IF                (yDeflectionHigh-hysteresisHighValue)>(yDeflectionLow-
hysteresisLowValue) THEN
          uHighValue := u0 +(uHighValue - u0)* 0.9; // uprava horni polohy rele
          ELSE
          uLowValue:= u0 +(uLowValue - u0)* 0.9; // uprava spodni polohy rele
       END_IF;
       gama  :=  MAX(ABS(uHighValue-u0),  ABS(u0-uLowValue))/MIN(ABS(uHighValue-
u0),ABS(u0-uLowValue)); // gama
    END_CASE;
         u:=uHighValue; // nabezna hrana
         tauTemp := 0.0; // vynulovani docasneho tau
         T1Temp :=T#0s; // vynulovani docasneho T1
         T2Temp:=T#0s; // vynulovani docasneho T2
         roTemp:=0.0; // vynulovani docasneho ro
         yDeflectionHighLast  :=  yDeflectionHigh;  // predchozi  hodnota  horni
vychylky
         yDeflectionHigh := 0.0;  // vynulovani horni vychylky
         yDeflectionLowLast  :=  yDeflectionLow;  //  predchozi  hodnota  spodni
vychylky
         yDeflectionLow := 0.0; // vynulovani spodni vychylky
         gama := MAX(ABS(uHighValue-u0), ABS(u0-uLowValue))/MIN(ABS(uHighValue-
u0),ABS(u0-uLowValue)); // gama
         ELSE
         yConditions :=0; // zmena CASE
         webOutputString:= 'Probiha identifikace soustavy'; // informace pro
uzivatele
         runningBlockCase :=80; // prechod do dalsiho CASE
      END_IF;
   END_IF;

////////////////////////////////////////////////////////////////////////////
////////////////////////////////// vzorkovani //////////////////////////////
////////////////////////////////////////////////////////////////////////////

  80: // Init

     runningBlockCase :=81; // prechod do dalsiho CASE
     gama  :=  MAX(ABS(uHighValue-u0),  ABS(u0-uLowValue))/MIN(ABS(uHighValue-
u0),ABS(u0-uLowValue)); //vypočtení stupně asymetrie relé
     u:=uHighValue; // nabezna
```

```
81: // Main

    IF y0-y > hysteresisLowValue THEN // mela by byt horni poloha rele
       IF u <> uHighValue THEN // neni nabezna
          IF ySamplingArrayDiff01 < deviation AND  ySamplingArrayDiff02 <
deviation AND ySamplingArrayDiff12 < deviation AND ySamplingArrayDiff01 > 0.0
AND  ySamplingArrayDiff02 >0.0 AND ySamplingArrayDiff12 >0.0 THEN // opoustaci
podminka
             runningBlockCase :=90; // prechod do dalsiho CASE
             RETURN;
          END_IF;
          T1 := SUB_DT_DT(trailingEdgeRtc,leadingEdgeRtc); // T1
          leadingEdgeRtc := GetRTC(); // cas nabezne
          samplingArrayColumnIndex := samplingArrayColumnIndex +1; //posunutí
polohy v pameti
                IF samplingArrayColumnIndex > 2 THEN  // uprava nesmyslneho
cisla sloupce pole
                   samplingArrayColumnIndex:=0;
                END_IF;
          samplingArrayLastIndexes[samplingArrayColumnIndex]:=0; //vynulování
          CASE samplingArrayColumnIndex OF //prirazeni prumernych odchylek
hodnot vystupu
               0:
                 ySamplingArrayDiff01:=
ySamplingArrayDiffA/(UINT_TO_REAL(samplingArrayLastIndexes[1])+1);
                 ySamplingArrayDiff02:=
ySamplingArrayDiffB/(UINT_TO_REAL(samplingArrayLastIndexes[2])+1);
               1:
                 ySamplingArrayDiff01:=
ySamplingArrayDiffA/(UINT_TO_REAL(samplingArrayLastIndexes[0])+1);
                 ySamplingArrayDiff12:=
ySamplingArrayDiffB/(UINT_TO_REAL(samplingArrayLastIndexes[2])+1);
               2:
                 ySamplingArrayDiff02:=
ySamplingArrayDiffA/(UINT_TO_REAL(samplingArrayLastIndexes[0])+1);
                 ySamplingArrayDiff12:=
ySamplingArrayDiffB/(UINT_TO_REAL(samplingArrayLastIndexes[1])+1);
          END_CASE;
          ySamplingArrayDiffA := 0.0; // vymazani sumace rozdilu
          ySamplingArrayDiffB := 0.0; // vymazani sumace rozdilu
          samplingArrayIndexOld := 1; // vraceni indexu na zacatek
       END_IF;
       u := uHighValue; // nabezna
    END_IF;
    IF y0-y <(-hysteresisHighValue) THEN // jsme nad mezi hystereze
       IF u <> uLowValue THEN // neni spodni poloha
          T2 := SUB_DT_DT(leadingEdgeRtc,trailingEdgeRtc); // T2
          trailingEdgeRtc := GetRTC(); // cas spadove hrany
          trailingEdgeIndex[samplingArrayColumnIndex]:=
REAL_TO_UINT(CEIL(TIME_TO_REAL(timeInPeriod)*0.001/samplingPeriodTimeSpan));
// index spadove hrany
       END_IF;
       u :=uLowValue; // spadova hrana
    END_IF;
    Tperiod :=T1 +T2; // perioda
    timeInPeriod  :=  SUB_DT_DT(GetRTC(),leadingEdgeRtc);  //  cas  od  pocatku
periody
    samplingArrayIndex:=
REAL_TO_UINT(CEIL(TIME_TO_REAL(timeInPeriod)*0.001/samplingPeriodTimeSpan));
// vypocet indexu pole pro ulozeni vzorku
    IF samplingArrayIndex < 2999 THEN
       IF  samplingArrayIndex  <>  samplingArrayIndexOld  THEN   //  cekani  na
spravny cas vzorkovani
          samplingArrayIndexOld  :=  samplingArrayIndex;  //  ulozeni  minuleho
indexu vzorkovani
          CASE samplingArrayColumnIndex OF
          0:
```

```
            IF samplingArrayIndex <= samplingArrayLastIndexes[1] THEN
               ySamplingArrayDiffA:=          ySamplingArrayDiffA          +
ABS(ySamplingArray[1,samplingArrayIndex]-y); //vypocet sumy odychlek vzorku od
aktualni hodnoty
            END_IF;
            IF samplingArrayIndex <= samplingArrayLastIndexes[2] THEN
               ySamplingArrayDiffB:=          ySamplingArrayDiffB          +
ABS(ySamplingArray[2,samplingArrayIndex]-y); //vypocet sumy odychlek vzorku od
aktualni hodnoty
            END_IF;
          1:
            IF samplingArrayIndex <= samplingArrayLastIndexes[0] THEN
               ySamplingArrayDiffA:=          ySamplingArrayDiffA          +
ABS(ySamplingArray[0,samplingArrayIndex]-y); //vypocet sumy odychlek vzorku od
aktualni hodnoty
            END_IF;
            IF samplingArrayIndex <= samplingArrayLastIndexes[2] THEN
               ySamplingArrayDiffB:=                      ySamplingArrayDiffB
+ABS(ySamplingArray[2,samplingArrayIndex]-y); //vypocet sumy odychlek vzorku od
aktualni hodnoty
            END_IF;
          2:
            IF samplingArrayIndex <= samplingArrayLastIndexes[0] THEN
               ySamplingArrayDiffA:=                      ySamplingArrayDiffA
+ABS(ySamplingArray[0,samplingArrayIndex]-y); //vypocet sumy odychlek vzorku od
aktualni hodnoty
            END_IF;
            IF samplingArrayIndex <= samplingArrayLastIndexes[1] THEN
               ySamplingArrayDiffB:=ySamplingArrayDiffB
+ABS(ySamplingArray[1,samplingArrayIndex]-y); //vypocet sumy odychlek vzorku od
aktualni hodnoty
            END_IF;
          END_CASE;
       END_IF;
       ySamplingArray[samplingArrayColumnIndex,samplingArrayIndex]:=y;
//uložit vzorek
       samplingArrayLastIndexes[samplingArrayColumnIndex]:=
samplingArrayIndex;  //ulozeni posledniho indexu
     END_IF;

////////////////////////////////////////////////////////////////////////////
///////////////// kontrola tau a výpočet parametrů modelu //////////////////
////////////////////////////////////////////////////////////////////////////

  90:
  //2: //určení časů, výpočet Half-period ratio a normalized time delay
    Tperiod:=
REAL_TO_TIME(UINT_TO_REAL(samplingArrayLastIndexes[samplingArrayColumnIndex])*
samplingPeriodTimeSpan*1000.0); // perioda kmitu
    T1:=
REAL_TO_TIME(UINT_TO_REAL(trailingEdgeIndex[samplingArrayColumnIndex])*samplin
gPeriodTimeSpan*1000.0); // T1
    T2 := Tperiod - T1; // T2
    ro:=
MAX(TIME_TO_REAL(T1),TIME_TO_REAL(T2))/MIN(TIME_TO_REAL(T1),TIME_TO_REAL(T2));
// ro
    tau := (gama - ro)/((gama-1.0)*((0.35*ro)+0.65)); // tau
    IF tau <= 1.0 AND tau >= 0.05 THEN //kontrola spravnosti tau
       runningBlockCase :=91; // prechod do dalsiho CASE
       webOutputString:= 'Probihaji vypocty parametru regulatoru'; // informace
pro uzivatele
       u := uMinValue; // nastaveni akcni veliciny na minimum
       ELSE
           coefRelay:= 0.9;
           IF (TIME_TO_REAL(T1)-TIME_TO_REAL(T2))<0.0 THEN
               uHighValue   :=   u0+((uHighValue-u0)*coefRelay);   //   prepocet
parametru rele
```

```
               gama         :=         MAX(ABS(uHighValue-u0),          ABS(u0-
uLowValue))/MIN(ABS(uHighValue-u0),ABS(u0-uLowValue)); // gama
            END_IF;
            IF (TIME_TO_REAL(T1)-TIME_TO_REAL(T2))>0.0 THEN  // gama
               uLowValue := u0+((uLowValue-u0)*coefRelay); // prepocet parametru
rele
               gama         :=         MAX(ABS(uHighValue-u0),          ABS(u0-
uLowValue))/MIN(ABS(uHighValue-u0),ABS(u0-uLowValue)); // gama
            END_IF;
       runningBlockCase :=80; // prechod do dalsiho CASE
       webOutputString := 'Mereni se opakuje, program prepocita meze rele'; //
informace pro uzivatele
       ySamplingArrayDiff01 := 0.0; // vynulovani
       ySamplingArrayDiff02 := 0.0; // vynulovani
       ySamplingArrayDiff12 := 0.0; // vynulovani
    END_IF;

  91: // Main

     divisionTdT :=tau/(1.0-tau); // podil Td a T
     tempReal:=
(UINT_TO_REAL(trailingEdgeIndex[samplingArrayColumnIndex])*samplingPeriodTimeS
pan); // pomocny vypocet
     tempReal2:=             ((hysteresisHighValue/ABS(K)))-ABS(uLowValue-u0)+
((ABS(uHighValue-u0)+ABS(uLowValue-u0))* EXP(divisionTdT )); // pomocny vypocet
     T          :=          tempReal/          LN(tempReal2/(ABS(uHighValue-u0)-
(hysteresisHighValue/ABS(K)))); // casova konstanta
     Td := T*divisionTdT; // dopravni zpozdeni
     runningBlockCase := 100; // prechod do dalsiho CASE

//////////////////////////////////////////////////////////////////////////////
////////////////////// priprava na dalsi identifikaci ////////////////////////
//////////////////////////////////////////////////////////////////////////////

  100: // Init
     runningBlockCase :=101; // prechod do dalsiho CASE

  101: // Main
     buttonIsPressed:= FALSE; // uvolneni tlacitka
     webOutputString := ''; // vymazani hlaseni pro uzivatele
     runningBlockCase :=00; // prechod pro dalsiho CASE

  END_CASE;

END_PROGRAM
```

```
// fbwebInputValidation

(* funkcni blok pro kontrolu zadaných parametru identifikace

input
minValue - minimalni pripustna hodnota akcni veliciny
maxValue - maximalni pripustna hodnota akcni veliciny
minOffset - vzdalenost bodu blizkeho pracovnimu bodu (rozdil akcnich velicin),
ovlivnuje urceni staticke citlivosti
value - akcni velicina v pracovnim bode

outut
isValid - ukazatel platnosti zadanych parametru
message - zprava uzivateli, aby vedel jak spravne upravit zadanée parametry
*)

FUNCTION_BLOCK webInputValidation
  VAR_INPUT
  minValue : REAL; // minimalni hodnota akcni veliciny
  maxValue : REAL; // maximalni hodnota akcni veliciny
  minOffset : REAL; // snizeni akcni veliciny
  value : REAL; // akcni velicina v pracovnim bode
  END_VAR
  VAR_OUTPUT
  isValid : BOOL := false; // priznak platnosti zadanych parametru
  message : STRING; // zprava uzivateli
  END_VAR
  VAR_IN_OUT
  END_VAR
  VAR
  END_VAR
  VAR_TEMP
  END_VAR

  IF (value + minOffset > minValue) AND value < maxValue THEN // kontrola
zadanych udaju
     isValid := true; // priznak splneni spravneho zadani parametru
  END_IF;
  IF minValue < 0 OR maxValue > 10 THEN // kontrola zadanych udaju
     message := 'horni a dolni mez musi lezet mezi 0 a 10'; // zprava
uzivateli
     isValid := false; // priznak nesplneni spravneho zadani parametru
  END_IF;
  IF minValue > maxValue THEN // kontrola zadanych udaju
     message := 'horni mez musi lezet nad dolni mezi'; // zprava uzivateli
     isValid := false; // priznak nesplneni spravneho zadani parametru
  END_IF;
  IF (value + minOffset < minValue) OR value > maxValue THEN // kontrola
zadanych udaju
     message := 'vstup musi lezet mezi dolni a horni mezi'; // zprava
uzivateli
     isValid := false; // priznak nesplneni spravneho zadani parametru
  END_IF;

END_FUNCTION_BLOCK
```

```
// fbstabilization

(* funkcni blok pro kontrolu ustaleni vystupu soustavy dle zmeny monotonnosti
pro casove prumery

input
vstup - vstup do fb, hodnota vystupu ze soustavy
cas_mereni - doba po kterou sbirame hodnoty vystupu ze soustavy pro vypocet
prumeru

output
ustaleno - vraci hodnotu TRUE pokud doslo ke zmene monotonosti casovych
prumeru, tedy pokud je soustava ustalena
*)

FUNCTION_BLOCK fbstabilization
  VAR_INPUT
  vstup : REAL; // vystup ze soustavy
  cas_mereni : TIME := T#5s; // trvani casovace
  END_VAR
  VAR_OUTPUT
  ustaleno : BOOL := FALSE; // indikator ustaleni
  END_VAR
  VAR_IN_OUT
  END_VAR
  VAR
   //kontrola ustaleni
   timer1 : TON; // casovac pro kontrolu ustaleni soustavy
   y1_k : REAL; // prumerna hodnota vystupu po dobu casovace
   y2_k : REAL; // prumerna hodnota vystupu po dobu casovace
   y3_k : REAL; // prumerna hodnota vystupu po dobu casovace
   y_suma_k : REAL; // suma vzorku vystupu soustavy po dobu casovace
   pocitadlo : REAL; // pocet vzorku vystupu soustavy po dobu casovace
   klid : INT; // case
  END_VAR
  VAR_TEMP
  END_VAR


CASE klid OF

////////////////////////////////////////////////////////////////////////////
0:
     ustaleno := FALSE; // negace ukazatele ustaleni
     IF timer1.Q = FALSE THEN // nedobehl casovac
        pocitadlo := pocitadlo +1.0; // citac vzorku
        y_suma_k := y_suma_k + vstup; // suma vzorku
     END_IF;
     timer1(IN:= TRUE, PT :=cas_mereni); // casovac
     IF timer1.Q THEN // casovac dobehl
        timer1(IN:= FALSE); // vynulovani casovace
        y1_k := y_suma_k/pocitadlo; // vypocet prumerne hodnoty regulovane
veliciny
        pocitadlo := 0.0; // vynulovani citace
        y_suma_k := 0.0; // vynulovani sumy vzorku regulovane veliciny
        klid := 1; // prechod do dalsiho CASE
     END_IF;
////////////////////////////////////////////////////////////////////////////
1:
     IF timer1.Q=FALSE THEN // nedobehl casovac
        pocitadlo := pocitadlo +1.0; // citac vzorku
        y_suma_k := y_suma_k + vstup; // suma vzorku
     END_IF;
     timer1(IN:= TRUE, PT :=cas_mereni); // casovac
     IF timer1.Q THEN // casovac dobehl
        timer1(IN:= FALSE); // vynulovani casovace
        y2_k := y_suma_k/pocitadlo; // vypocet prumerne hodnoty regulovane
veliciny
```

```
        pocitadlo := 0.0; // vynulovani citace
        y_suma_k := 0.0; // vynulovani sumy vzorku regulovane veliciny
        klid := 2 ; // prechod do dalsiho CASE
     END_IF;
////////////////////////////////////////////////////////////////////////////
2:
     IF timer1.Q=FALSE THEN // nedobehl casovac
        pocitadlo := pocitadlo +1.0; // citac vzorku
        y_suma_k := y_suma_k + vstup; // suma vzorku
     END_IF;
     timer1(IN:= TRUE, PT :=cas_mereni); // casovac
     IF timer1.Q THEN // casovac dobehl
        timer1(IN:= FALSE); // vynulovani casovace
        y3_k := y_suma_k/pocitadlo; // vypocet prumerne hodnoty regulovane
veliciny
        pocitadlo := 0.0; // vynulovani citace
        y_suma_k := 0.0; // vynulovani sumy vzorku regulovane veliciny
        klid := 3 ; // prechod do dalsiho CASE
     END_IF;
////////////////////////////////////////////////////////////////////////////
3:
     IF (y1_k > y2_k AND y2_k > y3_k) OR (y1_k < y2_k AND y2_k < y3_k) THEN //
porovnani prumernych hodnot regulovane veliciny
     y1_k := y2_k; // posun o misto v pameti
     y2_k := y3_k; // posun o misto v pameti
     klid := 2; // prechod do dalsiho CASE
     ELSE
     klid := 4; // prechod do dalsiho CASE
     y1_k := 0; // vynuovani prumerne hodnoty regulovane veliciny
     y2_k := 0; // vynuovani prumerne hodnoty regulovane veliciny
     y3_k := 0; // vynuovani prumerne hodnoty regulovane veliciny
     END_IF;
////////////////////////////////////////////////////////////////////////////
4:
ustaleno := TRUE; // nastaveni priznaku ustaleni soustavy
klid:= 0; // navrat do prvniho CASE
END_CASE;
END_FUNCTION_BLOCK
```

```
// fbparametrySOTD

(* Výpočet paramerů SOTD modelu dle práce Alternative Identification Method
using Biased Relay Feedbeck autora M. Hofreitera

                  K_p               -(T_d*s)
M(s)= --------------------------- e
            (a_2*s^2+a_1*s+1)

inputs
omega_1 - uhlová frekvence 1
bod_1_real - reálná souřadnice bodu frekvenční charakteristiky pro frekvenci
omega_1
bod_1_imag - imaginární souřadnice bodu frekvenční charakteristiky pro frekvenci
omega_1
omega_2 - uhlová frekvence 2
bod_2_real - reálná souřadnice bodu frekvenční charakteristiky pro frekvenci
omega_1
bod_2_imag - imaginární souřadnice bodu frekvenční charakteristiky pro frekvenci
omega_1
K_s - statická citlivost soustavy

outputs
a_2 - časová konstanta modelu
a_1 - časová konstanta modelu
T_d - dopravní zpoždění modelu
K_p - statická citlivost modelu
*)

FUNCTION_BLOCK fbparametrySOTD

  VAR_INPUT
  omega_1 : REAL;
  omega_2 : REAL;
  bod_1 : COMPLEX;
  bod_2 : COMPLEX;
  K_s : REAL;
  END_VAR
  VAR_OUTPUT
  a_2 : REAL;
  a_1 : REAL;
  T_d : REAL;
  K_p : REAL;
  END_VAR
  VAR_IN_OUT
  END_VAR

  VAR

  //pomocné proměné
  bod_1_velikost : REAL; // absolutni hodnota z komplexniho cisla
  bod_2_velikost : REAL; // absolutni hodnota z komplexniho cisla
  pod_odmocninou_1 : REAL; // vypocet odmocniny
  pod_odmocninou_2 : REAL; // vypocet odmocniny
  zlomek_2 : REAL; // vypocet zlomku
  zlomek_1 : REAL; // vypocet zlomku
  argument_1_sum_1 : REAL; // argument pri vypoctu Td
  argument_2_sum_1 : REAL; // argument pri vypoctu Td
  argument_1_sum_2 : REAL; // argument pri vypoctu Td
  argument_2_sum_2 : REAL; // argument pri vypoctu Td
  complex_1 : COMPLEX := ( re := 0.1,  im := 0.0); // 1 v komplexnich cislech
  jmenovatel_1 : COMPLEX; // komplexni jmenovatel
  jmenovatel_2 : COMPLEX; // komplexni jmenovatel  END_VAR
  VAR_TEMP
  END_VAR

  K_p := K_s; // prirazeni staticke citlivosti
  bod_1_velikost := CABS(bod_1); // vypocet absolutni hodnoty
```

```
  bod_2_velikost := CABS(bod_2); // vypocet absolutni hodnoty
  zlomek_1 := (K_p*K_p)/(bod_2_velikost*bod_2_velikost); // mezivypocet
  zlomek_2 := (4*K_p*K_p)/(bod_1_velikost*bod_1_velikost); // mezivypocet
  pod_odmocninou_1 := (1/3)*(zlomek_1 - zlomek_2 +3); // mezivypocet
  a_2 := (1/(2*omega_1*omega_1))*SQRT(pod_odmocninou_1); // casova konstanta
  pod_odmocninou_2          :=          (zlomek_2/4)-(1-a_2*omega_1*omega_1)*(1-
a_2*omega_1*omega_1); // mezivypocet
  a_1 := (1/omega_1)*SQRT(pod_odmocninou_2); // casova konstanta
  jmenovatel_1.re := -a_2*omega_1*omega_1+1; // realna slozka jmenovatele
  jmenovatel_1.im := a_1*omega_1; // imaginarni slozka jmenovatele
  jmenovatel_2.re := -a_2*omega_2*omega_2+1; // realna slozka jmenovatele
  jmenovatel_2.im := a_1*omega_2; // realna slozka jmenovatele
  argument_1_sum_1:=  CARG(CDIV(X:=complex_1,  Y:=jmenovatel_1));  //  vypocet
argumentu konplexniho cisla
  argument_2_sum_1:= CARG(bod_1); // vypocet argumentu konplexniho cisla
  argument_1_sum_2:=  CARG(CDIV(X:=complex_1,  Y:=jmenovatel_2));  //  vypocet
argumentu konplexniho cisla
  argument_2_sum_2:= CARG(bod_2); // vypocet argumentu konplexniho cisla
  T_d:=     0.5*((argument_1_sum_1-argument_2_sum_1)/omega_1+(argument_1_sum_2-
argument_2_sum_2)/omega_2); // dopravni zpozdeni
END_FUNCTION_BLOCK
```

```
// fbRelayFeedbackIdentification

(*funkcni blok releove zpetnovazebni identifikace dle práce Alternative
Identification Method using Biased Relay Feedbeck autora M. Hofreitera

             K_p               -(T_d*s)
M(s)=------------------------- e
          (a_2*s^2+a_1*s+1)

inputs
y - regulovana velicina
uMaxValue - maximalni hodnota vstupu
uMinValue - minimalni hodnota vstupu
u0 - vstup v pracovnim bode

outputs
u - akcni velicina
sotdA1 - casova konstanta modelu SOTD
sotdA2 - druha casova konstanta modelu SOTD
sotdK - staticka citlivost modelu SOTD
sotdTau - dopravni zpozdeni modelu SOTD
completed - priznak, ze je identifikace dokoncena
webOutputString - vystup na web, informace uzivateli
*)

FUNCTION_BLOCK fbRelayFeedbackIdentification

(*funkcni blok releove zpetnovazebni identifikace
*)
  VAR_INPUT
  y:REAL; // regulovana velicina
  uMaxValue : REAL := 10;  // maximalni hodnota vstupu
  uMinValue : REAL := 0;  // minimalni hodnota vstupu
  u0 : REAL := 3;    // vstup v pracovnim bode
  END_VAR
  VAR_OUTPUT
  u:REAL; // akcni velicina
  sotdA1 : REAL;   // casova konstanta modelu SOTD
  sotdA2 : REAL;   // druha casova konstanta modelu SOTD
  sotdK : REAL;   // staticka citlivost v modelu SOTD
  sotdTau :REAL;   // dopravni zpozdeni v modelu SOTD
  completed :BOOL := FALSE; // priznak jestli je identifikace dokoncena
  webOutputString : STRING;  // vystup na web
  END_VAR

VAR_IN_OUT
  END_VAR
  VAR
  ySamplingArray  :  ARRAY  [0..2,  0..1300]  OF  REAL;  //  vzorkovani  prubehu
regulovane veliciny
  uSamplingArray : ARRAY [0..2, 0..1300] OF REAL; //vzorkovani prubehu akcni
veliciny
  samplingArrayLastIndexes : ARRAY [0..2] OF UINT;  //pocty vzorku
  samplingArrayColumnIndex   : USINT := 2; //index používaného sloupce
  yK1 : REAL; // regulovana veličina v bode blizkem pracovnímu bodu
  yK2 : REAL; // regulovana veličina v bode blizkem pracovnímu bodu
  uK1 : REAL; // akcni veličina v bode blizkem pracovnímu bodu
  uK2 : REAL; // akcni veličina v bode blizkem pracovnímu bodu
  trailingEdgeIndex : ARRAY [0..2] OF UINT;  //index na kterém došlo k přepnutí
relé (u)
  currentLastIndex : UINT; // minuly index
  leadingEdgeCounter : INT;  // pocet nabeznych hran
  runningBlockCase : INT :=0;  // aktualni blok kodu
  tempCounter : INT :=0;  // pocet vzorku po detekci ustaleni (yk1, yk2)
  tempCounter2 : INT :=0;  // citac
  trailingEdgeCounter : INT;  // pocet sestupnych hran
  indexIntegral   : UINT;  // index pri integraci
  indexShiftIntegral : UINT;  // posunuty index pri integraci (posuv = Tperiod/2)
```

```
  samplingArrayColumnNeighbour1Index : UINT;   // index sousedního sloupce pole
  samplingArrayColumnNeighbour2Index : UINT;    // index sousedního sloupce pole
  samplingArrayIndex   : UINT;   // index pri vzorkovani
  samplingArrayIndexOld : UINT;  // puvodni index pri vzorkovani z predchoziho
pruchodu
  buttonIsPressed : BOOL :=1;   // input z buttonu
  tempBool : BOOL := false;  // docasna bool hodnota
  tempBool2 : BOOL := false;  // docasna bool hodnota
  samplingExit : bool := false;   // priznak, zda opustit vzorkovani
  hysteresisHighValue : REAL;  // horni mez hystereze
  hysteresisLowValue : REAL;  // dolni mez hystereze
  integralExponent1 : REAL;  // exponent ve vzorci s integralem (realny)
  integralExponent2 : REAL;  // exponent ve druhem vzorci s integralem (realny)
  K :REAL;  // staticka citlivost
  noiseMax : REAL;  // maximalni sum
  omega1 : REAL;   // uhlova rychlost ve vzorci s integralem
  omega2 : REAL;   // uhlova rychlost ve druhem vzorci s integralem
  samplingDelayMax  :  REAL;  //  maximalni  zpozdeni  pri  vzorkovani  pole  (0;
samplingPeriodTimeSpan)
  samplingDelaySum : REAL;  // suma vsech zpozdeni pri vzorkovani pole
  samplingDelaySumMax : REAL;  // maximalni suma vsech zpozdeni pri samplovani
poli
  samplingPeriodTimeSpan :  REAL;   // vzorkovací perioda integrátoru
  tauIndexIntegral : REAL;   // cas pri integraci
  tempReal : REAL;  // docasna promenna pro real
  tempReal4 : REAL;   // docasna promenna pro real
  tempSum : REAL;   // pomocna promenna pro vypocet prumeru vystupu
  timeDiff : REAL;    // pomocna promenna pro urcovani rozdilu casu
  trafficDelay : REAL;   // dopravni zpozdeni
  T1 : REAL;   // cas pro rele v horni poloze
  T2 : REAL;   // cas pro rele ve spodni poloze
  Tperiod : REAL;   // delka periody [ms]
  uHighValue : REAL;  // hodnota vstupu pri horni poloze rele
  uLowValue : REAL;   // hodnota vstupu pri spodni poloze rele
  y0 : REAL;  // vystup v pracovnim bode
  ySamplingArrayDiffSum : REAL; // suma rozdilu mezi namerenymi hodnotami Y v
poslednich trech pruchodech
  ySamplingArraySum : REAl;  // suma namerenych hodnot v poli Y
  G1   :  COMPLEX;  // prvni bod Nyquistovy frekvencni charakteristiky v komplexni
rovine
  G2   :  COMPLEX;  // druhy bod Nyquistovy frekvencni charakteristiky v komplexni
rovine
  integralComplexExponent1 : COMPLEX;  // komplexni exponent v prvnim integralu
  integralComplexExponent2 : COMPLEX;  // komplexni exponent ve druhem integralu
  integralU1 : COMPLEX;  // vysledek celeho integralu ve jmenovateli 1. vzorce
  integralU2 : COMPLEX;  // vysledek celeho integralu ve jmenovateli 2. vzorce
  integralY1 : COMPLEX;  // vysledek celeho integralu v citateli 1. vzorce
  integralY2 : COMPLEX;  // vysledek celeho integralu v citateli 2. vzorce
  uSampleComplex1 : COMPLEX;  // vzorek v komplexni rovine ve jmenovateli 1.
vzorce
  uSampleComplex2 : COMPLEX;  // vzorek v komplexni rovine ve jmenovateli 2.
vzorce
  ySampleComplex1 : COMPLEX;  // vzorek v komplexni rovine v citateli 1. vzorce
  ySampleComplex2 : COMPLEX;  // vzorek v komplexni rovine v citateli 2. vzorce
  leadingEdgeRtc : DATE_AND_TIME;  // cas nabezne hrany
  trailingEdgeRtc : DATE_AND_TIME;  // cas sestupne hrany
  tempDateTime : DATE_AND_TIME;  // docasna promenna pro datetime
  T0 : DATE_AND_TIME;  // pocatek vzorkovaci periody
  timer : TON;  // casovac
  timerForDelay : TON;  //casovac pro zpozdeni nabezne a spadove hrany
  fb_webInputValidation  :  webInputValidation;  //funkci  blok  pro  kontrolu
zadanych udaju
  fb_stabilization : fbstabilization;  //funkcni blok pro kontrolu ustaleni
  fb_parametrySOTD : fbparametrySOTD;  //funkcni blok vypoctu parametru modelu
  END_VAR
  VAR_TEMP
  END_VAR
```

```
  // Urceni parametru modelu ze tri bodu Nyquistovy frekvencni charakteristiky
s pridanym dopravnim zpozdenim

  CASE runningBlockCase OF

////////////////////////////////////////////////////////////////////////////
////////////// nastaveni parametru a cekani na buttonIsDisabled /////////////////
////////////////////////////////////////////////////////////////////////////

  00: // Init
    runningBlockCase := 01; // prechod do dalsiho CASE

  01: // Main
    IF buttonIsPressed = TRUE THEN // kontrola stisknuti tlacitka
      fb_webInputValidation(minValue:= uMinValue, minOffset := -0.5, maxValue:=
uMaxValue,  value:=u0,  isValid  =>  tempBool,  message  =>  webOutputString);
//volání podprogramnu pro kontrolu zadaných parametru
      IF tempBool = FALSE THEN // kdyz jsou spatne zadane udaje
        buttonIsPressed := FALSE; //uvolněni tlacitka
      ELSE
        runningBlockCase := 10; // prechod do dalsiho CASE
        completed := FALSE; // priznak ze neni hotovo
      END_IF;
    END_IF;

////////////////////////////////////////////////////////////////////////////
//////////////////////////////  cekani na ustaleni  ////////////////////////////
////////////////////////////////////////////////////////////////////////////

  10: // Init
    u:=u0-0.5;    // nastavení vstupního napití pod pracovní bod, kvuli ureení
statické citlivosti
    runningBlockCase := 11; // prechod do dalsiho CASE

  11: // Main
    fb_stabilization(vstup:=y,  cas_mereni:=  T#10s,  ustaleno  =>  tempBool);  //
cekani na ustaleni vystupu
    IF tempBool = TRUE THEN // kdyz je vystup ustaleny
      runningBlockCase := 20; // prechod do dalsiho CASE
    END_IF;

////////////////////////////////////////////////////////////////////////////
//////////////////////////////  urceni u1 a y1  ////////////////////////////////
////////////////////////////////////////////////////////////////////////////

  20: // Init
    tempCounter := 0; //vynulovani citace
    tempSum := 0; //vynulovani promenne
    runningBlockCase := 21; // prechod do dalsiho CASE

  21: // Main
    timer(IN:=TRUE, PT:=T#1s); // casovac pro urceni prumerne hodnoty vystupu v
bode 1 pro urceni staticke citlivosti
    tempSum := tempSum+y;  // suma vystupu po dobu behu casovace
    tempCounter := tempCounter+1; // pocet sectenych vystupu po dobu casovace
    IF timer.Q THEN // po uplynutí easovaee
      timer(IN:=FALSE); // vynulování casovace
      yK1  :=  tempSum/INT_TO_REAL(tempCounter);  //  vypocet  prumerne  hodnoty
vystupu
      uK1 := u0 - 0.5; // hodnota akcni veliciny
      runningBlockCase := 30; // prechod do dalsiho CASE
    END_IF;

////////////////////////////////////////////////////////////////////////////
//////////////////////////////  cekani na ustaleni  ////////////////////////////
////////////////////////////////////////////////////////////////////////////
```

```
  30: // Init
    u:=u0; //nastavení akcni veliciny na zadanou hodnotu, tj. pracovni bod
    runningBlockCase := 31; // prechod do dalsiho CASE

  31: // Main
    fb_stabilization(vstup:=y, cas_mereni:= T#10s, ustaleno => tempBool); //
cekani na ustaleni vystupu
    IF tempBool = TRUE THEN // kdyz je ustaleny vystup
       runningBlockCase := 40; // prechod do dalsi faze programu
    END_IF;

////////////////////////////////////////////////////////////////////////////
////////////////////////////// urceni u2 a y2 //////////////////////////////
////////////////////////////////////////////////////////////////////////////

  40: // Init
    tempSum := 0; // vynulovani promenne
    tempCounter := 0; // vynulovyni citace
    runningBlockCase := 41; // prechod do dalsiho CASE

  41: // Main
    timer(IN:=TRUE, PT:=T#1s); // casovac pro vypocteni prumerne hodnoty vystupu
    tempSum := tempSum+y; // suma vzorku vystupu
    tempCounter := tempCounter+1; //pocet sectenych vystupu
    IF timer.Q THEN // když casovac dobehne
      timer(IN:=FALSE); // nulovani casovace
      yK2 := tempSum/INT_TO_REAL(tempCounter); // vypocet prumerneho vystupu v
bode pro urceni staticke citlivosti
      uK2 := u0; //akcni velicina v bode pro urceni staticke citlivosti
      y0:=yK2; //regulovana velicina v pracovnim bode
      K:=(yK2-yK1)/(uK2-uK1); //vypocet hodnoty statické citlivosti
      runningBlockCase:=50; // prechod do dalsiho CASE
    END_IF;

////////////////////////////////////////////////////////////////////////////
////////////////////////////// mereni sumu /////////////////////////////////
////////////////////////////////////////////////////////////////////////////

  50: // Init
    noiseMax := 0; //vynulování nejvyšší hodnoty výstupu
    runningBlockCase:=51; // prechod do dalsiho CASE

  51: // Main
    timer(IN:=TRUE, PT:=T#10s); // casovac po ktery se meri sum
    IF timer.Q = FALSE THEN // kdyz casovac nedobehl
      IF ABS(y0-y)>noiseMax THEN // kdyz absolutní hodnota sumu je vetsi nez
nejvetsi ulozena
        noiseMax:=ABS(y0-y); // ulozeni velikosti sumu z aktualni absoulutni
hodnoty sumu
      END_IF;
    ELSE
      timer(IN:=FALSE); // vynulovani casovace
      hysteresisHighValue:=noiseMax*8; // urceni mezi hystereze rele z velikosti
sumu
      hysteresisLowValue:=noiseMax*10; // urceni mezi hystereze rele z velikosti
sumu
      runningBlockCase:=70; // prechod do dalsiho CASE
    END_IF;

  (* // uprava pro pouziti simulace soustavy
    noiseMax:=0.2;
    hysteresisHighValue:=noiseMax*6;
    hysteresisLowValue:=noiseMax*10;
    runningBlockCase :=70;
  *)

////////////////////////////////////////////////////////////////////////////
//////////////////////////////  urceni periody  ////////////////////////////
```

```
////////////////////////////////////////////////////////////////////////////

  70: // Init
    uHighValue := u0 + 5; // urceni horni polohy rele
    IF uHighValue > uMaxValue THEN // pokud horni poloha rele je vyssi nez
maximalni
      uHighValue:= uMaxValue; // uprava horni hodnoty rele
    END_IF; // horni mez rele

    uLowValue:= u0 - 3; // dolní poloha rele
    IF uLowValue < uMinValue THEN // kdyz je spodni poloha relé mensi nez
minimalni
      uLowValue:= uMinValue; // uprava spodni polohy rele
    END_IF;
    u:= uHighValue; // nastaveni akcni veliciny na horni mez rele
    trailingEdgeCounter:=0; // citac spadovych hran
    leadingEdgeCounter:=1; // citac nabeznych hran
    tempBool := FALSE; // ukazatel stejne delky casu T1
    tempBool2 := FALSE; // ukazatel stejne delky casu T2
    T1 := 1; // nastaveni casu T1 na nenulovou hodnotu
    T2 := 1; // nastaveni casu T2 na nenulovou hodnotu
    leadingEdgeRtc := getRTC(); // cas nabezne hrany
    trailingEdgeRtc := getRTC(); // cas spadove hrany
    runningBlockCase := 71; // prechod do dalsiho CASE

  71: // Main
  IF y>y0+hysteresisHighValue AND u=uHighValue THEN // kdyz je cas na sestupnou
hranu a jeste k ni nedoslo
    u:= uLowValue; // dolni mez rale
    tempDateTime:= getRTC(); // ulozeni aktualniho casu
    trailingEdgeCounter := trailingEdgeCounter + 1; // citac spadovych hran
    trailingEdgeRtc := tempDateTime; // cas spadove hrany
    tempReal   :=   (TIME_TO_REAL(SUB_DT_DT(tempDateTime,leadingEdgeRtc)));   //
aktualni cas T1
    IF abs(1.0 - tempReal/T1) < 0.03 THEN // kdyz je rozdil mezi soucasnym a
minulym T1 rozdil mensi nez 3 procenta
      tempBool := TRUE; // ukazatel splneni podminky
    ELSE
      tempBool := FALSE; // ukazatel na nesplneni podminky
    END_IF;
    T1 := tempReal; // prepsani casu T1
  END_IF;
  IF y<y0-hysteresisLowValue AND u=uLowValue THEN // kdyz ma byt nabezna hrana
a jeste k ni nedoslo
    u:= uHighValue; // nastavi se nabezna hrana
    tempDateTime:= getRTC(); // cas nabezne hrany
    leadingEdgeCounter := leadingEdgeCounter + 1; // citac nabeznych hran
    leadingEdgeRtc := tempDateTime; // cas nabezne hrany
    tempReal   :=   (TIME_TO_REAL(SUB_DT_DT(tempDateTime,trailingEdgeRtc)));   //
vypocet casu T2
    IF abs(1.0 - tempReal/T2) < 0.03 THEN // kdyz je rozdil mezi soucasnym a
minulym T2 rozdil mensi nez 3 procenta
      tempBool2 := TRUE; // ukazatel splneni podminky
    ELSE
      tempBool2 := FALSE; // ukazatel nesplneni podminky
    END_IF;
    T2 := tempReal; // cas T2
  END_IF;
  IF tempBool AND tempBool2 AND u=uHighValue THEN // kdyz jsou splneny podminky
a rele je nahore
    Tperiod := T1+T2; // vypocet periody
    TrafficDelay := Tperiod/2;
    runningBlockCase :=80; // prechod do dalsiho CASE
  END_IF;


////////////////////////////////////////////////////////////////////////////
//////////////////////////////  vzorkovani  ////////////////////////////////
```

```
////////////////////////////////////////////////////////////////////////////

  80: // Init
    IF T1 > T2 THEN // kontrola splneni podminky
      webOutputString := 'T1 je vetsi nez T2, sprav si to';   // hlaseni o
nesplneni podminky
      runningBlockCase := 1000; // prechod do dalsiho CASE
    END_IF;
    samplingArrayIndexOld := 1000; // nastaveni indexu na nesmyslny
    samplingArrayColumnIndex := 0; // nastaveni sloupce pole
    samplingArrayColumnNeighbour1Index := 1; // nastaveni sousedniho sloupce
    samplingArrayColumnNeighbour2Index := 2; // nastaveni druheho sousedniho
sloupce
    ySamplingArrayDiffSum := 0; // nulovani souctu rozdilu mezi sloupci
    ySamplingArraySum := 0; // soucet ulozenych prvku
    samplingArrayLastIndexes[0] := 1000; // nastaveni indexu na nesmyslny
    samplingArrayLastIndexes[1] := 1000; // nastaveni indexu na nesmyslny
    samplingArrayLastIndexes[2] := 1000; // nastaveni indexu na nesmyslny
    tempCounter := 0; // citac
    tempCounter2 := 0; // citac
    samplingDelayMax := 0; // nulovani maximalniho zpozdeni
    samplingDelaySum := 0; // nulovani souctu zpozdeni
    samplingDelaySumMax := 0; // nulovani maximalniho celkoveho souctu zpozdeni
    samplingPeriodTimeSpan := Tperiod / 900.0; // vzorkovaci perioda [ms], ale
opravdu! [ms]!!
    samplingPeriodTimeSpan  :=  10*CEIL(samplingPeriodTimeSpan/10);  //  uprava
vzorkovaci periody
    IF (samplingPeriodTimeSpan < 20) THEN  // kdyz je vzorkovaci perioda mensi
nez 20ms
      samplingPeriodTimeSpan := 20; // vzorkovaci perioda 20ms
    END_IF;
    T0:=GetRTC(); // aktualni cas
    runningBlockCase := 81; // prechod do dalsiho CASE

  81: // Main
    IF y>y0+hysteresisHighValue AND u=uHighValue THEN // kdyz je cas na sestupnou
hranu
      timerForDelay(IN := TRUE, PT := REAL_TO_TIME(trafficDelay)); // zpozdeni
sestupne hrany
      IF timerForDelay.Q = TRUE THEN  // kdyz casovac dobehne
        u:= uLowValue; // sestupna hrana
        tempCounter2 := tempCounter2+1; //citac
        timerForDelay(IN := FALSE); // vynulovani casovace
        tempDateTime:= getRTC();  // aktualni cas
        trailingEdgeRtc := tempDateTime; // cas spadove hrany
        tempReal := (TIME_TO_REAL(SUB_DT_DT(tempDateTime,leadingEdgeRtc))); //
T1
        T1 := tempReal; // T1
      END_IF;
    END_IF;
    IF y<y0-hysteresisLowValue AND u=uLowValue THEN // cas na nabeznou hranu
      timerForDelay(IN := TRUE, PT := REAL_TO_TIME(trafficDelay)); // zpozdeni
nabezne hrany
      IF timerForDelay.Q = TRUE THEN // dobehl casovac
        u:= uHighValue; // nabezna hrana
        timerForDelay(IN := FALSE); // vynulovani casovace
        tempDateTime:= getRTC(); // aktualni cas
        leadingEdgeRtc := tempDateTime; //cas nabezne hrany
        tempReal := (TIME_TO_REAL(SUB_DT_DT(tempDateTime,trailingEdgeRtc))); //
T2
        T2 := tempReal; // T2
        T0:= getRTC(); // aktualni cas
        IF (samplingDelaySum > samplingDelaySumMax) THEN // kontrola hodnoty
sumy zpozdeni
          samplingDelaySumMax := samplingDelaySum; // prepsani hodnoty sumy
zpozdeni
        END_IF;
        samplingDelaySum := 0; // vynulovani sumy zpozdeni
```

```
        IF tempCounter2 > 3 AND ySamplingArrayDiffSum/ySamplingArraySum < 0.06
THEN  // kontrola opousteci podminky
          samplingExit := TRUE; // ukazatel ukonceni vzorovani
          runningBlockCase := 90; // prechod do dalsiho CASE
        ELSE
          samplingArrayColumnIndex := samplingArrayColumnIndex + 1; // prechod
do dalsiho sloupce
        IF samplingArrayColumnIndex > 2 THEN  // kontrola cisla sloupce
          samplingArrayColumnIndex := 0; // uprava cisla sloupce
        END_IF;
        samplingArrayLastIndexes[samplingArrayColumnIndex] := 0; // vynulovani
cisla posledniho vzorku
        CASE samplingArrayColumnIndex OF  // upravy sloupcu sousedu
        0 :
          samplingArrayColumnNeighbour1Index := 1;
          samplingArrayColumnNeighbour2Index := 2;
        1 :
          samplingArrayColumnNeighbour1Index := 0;
          samplingArrayColumnNeighbour2Index := 2;
        2:
          samplingArrayColumnNeighbour1Index := 0;
          samplingArrayColumnNeighbour2Index := 1;
        END_CASE;
        ySamplingArraySum := 0; // vynulovani souctu vzorku
        ySamplingArrayDiffSum := 0; // vynulovani souctu rozdilu vzorku
      END_IF;
    END_IF;
    END_IF;
    IF samplingExit = FALSE THEN  // kdyz neni ukonceno vzorkovani
      timeDiff  :=  TIME_TO_REAL(SUB_DT_DT(GetRTC(),T0));  //  cas  od  pocatku
periody
      samplingArrayIndex:=            REAL_TO_UINT(FLOOR(timeDiff            /
samplingPeriodTimeSpan)); // index vzorku
      IF (samplingArrayIndex <> samplingArrayIndexOld) THEN // kdyz mame novy
index vzorku
        ySamplingArray[samplingArrayColumnIndex,samplingArrayIndex]  :=  y;  //
ulozeni vzorku regulovane veliciny
        uSamplingArray[samplingArrayColumnIndex,samplingArrayIndex]  :=  u;  //
ulozeni vzorku akcni veliciny
        samplingArrayLastIndexes[samplingArrayColumnIndex]:=
samplingArrayIndex; // prepsani posledniho indexu
        IF   samplingArrayLastIndexes[samplingArrayColumnNeighbour1Index]   >=
samplingArrayIndex THEN // pokud nebyl presazen pocet vzorku v sousednim sloupci
          ySamplingArrayDiffSum:=      ySamplingArrayDiffSum      +      abs(y-
ySamplingArray[samplingArrayColumnNeighbour1Index   ,samplingArrayIndex]);   //
prepocet sumy rozdilu
        END_IF;
        IF   samplingArrayLastIndexes[samplingArrayColumnNeighbour2Index]   >=
samplingArrayIndex THEN // pokud nebyl presazen pocet vzorku v sousednim sloupci
          ySamplingArrayDiffSum:=      ySamplingArrayDiffSum      +      abs(y-
ySamplingArray[samplingArrayColumnNeighbour2Index   ,samplingArrayIndex]);   //
prepocet sumy rozdilu
        END_IF;
        ySamplingArraySum := ySamplingArraySum + abs(y); // suma ulozenych prvku
        IF samplingArrayIndex = 0 THEN // kdyz je pocatek periody
          tempReal4 := timeDiff; // rozdil casu
        ELSE
          tempReal4  :=  timeDiff  -  (UINT_TO_REAL(samplingArrayIndexOld  +  1)  *
samplingPeriodTimeSpan); // rozdil casu
        END_IF;
        IF (tempReal4 > samplingDelayMax) THEN  // kontrola maximalniho zpozdeni
          samplingDelayMax := tempReal4; // maximalni zpozdeni
        END_IF;
        samplingDelaySum  :=  samplingDelaySum  +  tempReal4;  //  suma  zpozdeni
vzorkovani
        IF    (samplingArrayIndex    <>    0    AND    samplingArrayIndex    <>
samplingArrayIndexOld + 1) THEN
          tempCounter := tempCounter + 1; // citac vzorku
```

```
        END_IF;
        samplingArrayIndexOld := samplingArrayIndex; // minuly index
      END_IF;
    END_IF;

////////////////////////////////////////////////////////////////////////////
///////////////////////////// vypocet integralu ////////////////////////////
////////////////////////////////////////////////////////////////////////////

  90: // Init
    Tperiod := T1+ T2; // vypocet periody
    omega1   := MATH.PI*2 / (Tperiod * 0.001); // frekvence identifikace
    omega2  := 2.0*omega1; // dvojnasobek frekvence identifikace
    indexIntegral:=0; // index prvku pro integraci
    indexShiftIntegral:=
(samplingArrayLastIndexes[samplingArrayColumnIndex]/2); // posunuty index prvku
pro integraci
    currentLastIndex:=  samplingArrayLastIndexes[samplingArrayColumnIndex];  //
posledni index vzorku
    u :=uMinValue; // minimalizovani hodnoty akcni veliciny
    runningBlockCase := 91; // prechod do dalsiho CASE

  91: // Main
      IF  indexShiftIntegral  >  currentLastIndex  THEN  //  kontrola  posunuteho
indexu v integralu
        indexShiftIntegral := indexShiftIntegral - currentLastIndex; // uprava
posunuteho indexu v integralu
      END_IF;
      tauIndexIntegral    := UINT_TO_REAL(indexIntegral)*samplingPeriodTimeSpan
/ 1000; // cas v integralu jako desetiné císlo v sekundách
      integralExponent1:= omega1*tauIndexIntegral; // exponent integralu
      integralExponent2:= omega2*tauIndexIntegral; // exponent integralu
      integralComplexExponent1.re := 0; // realna slozka exponentu v integralu
      integralComplexExponent1.im := -integralExponent1; // imaginarni slozka
exponentu v integralu
      uSampleComplex1.im:=0; // imaginarni slozka vzorku akci veliciny
      uSampleComplex1.re:=
(uSamplingArray[samplingArrayColumnIndex,indexIntegral]);  //  realna  slozka
vzorku akci veliciny
      ySampleComplex1.im:=0; // imaginarni slozka vzorku regulovane veliciny
      ySampleComplex1.re:=
(ySamplingArray[samplingArrayColumnIndex,indexIntegral]);  //  realna  slozka
vzorku regulovane veliciny
      integralComplexExponent2.re := 0; // relna slozka exponentu v integralu
      integralComplexExponent2.im := -integralExponent2; // imaginarni slozka
exponentu v integralu
      uSampleComplex2.im:=0; // imaginarni slozka vzorku akci veliciny
      uSampleComplex2.re:=
(uSamplingArray[samplingArrayColumnIndex,indexIntegral])+(uSamplingArray[sampl
ingArrayColumnIndex,indexShiftIntegral]); // realna slozka vzorku akci veliciny
      ySampleComplex2.im:=0; // imaginarni slozka vzorku regulovane veliciny
      ySampleComplex2.re:=
(ySamplingArray[samplingArrayColumnIndex,indexIntegral])+(ySamplingArray[sampl
ingArrayColumnIndex,indexShiftIntegral]); // realna slozka vzorku regulovane
veliciny
      integralU1 := CADD(X := integralU1, Y :=(CMUL(X:=uSampleComplex1, Y:=
CEXP(X :=integralComplexExponent1)))); // jmenovatel zlomku pro vypocet druheho
bodu
      integralY1 := CADD(X := integralY1, Y :=(CMUL(X:=ySampleComplex1, Y:=
CEXP(X :=integralComplexExponent1)))); // citatel zlomku pro vypocet druheho
bodu
      integralU2 := CADD(X := integralU2, Y :=(CMUL(X:=uSampleComplex2, Y:=
CEXP(X :=integralComplexExponent2)))); // jmenovatel zlomku pro vypocet tretiho
bodu
      integralY2 := CADD(X := integralY2, Y :=(CMUL(X:=ySampleComplex2, Y:=
CEXP(X :=integralComplexExponent2)))); // citatel zlomku pro vypocet tretiho
bodu
      indexIntegral := indexIntegral + 1; // navyseni indexu v integralu
```

```
      indexShiftIntegral := indexShiftIntegral +1; // navyseni posunuteho indexu
v integralu
      IF indexIntegral = currentLastIndex THEN //konec intervalu
        runningBlockCase := 100; // prechod do dalsiho CASE
      END_IF;

////////////////////////////////////////////////////////////////////////////
///////////////////////////// vypocet G1 a G2  /////////////////////////////
////////////////////////////////////////////////////////////////////////////

  100: // Init
    runningBlockCase := 101; // prechod do dalsiho CASE

  101: // Main
    G1 := CDIV(X := integralY1, Y := integralU1); // druhy bod
    G2 := CDIV(X := integralY2, Y := integralU2); // treti bod
    runningBlockCase := 110; // prechod do dalsiho CASE

////////////////////////////////////////////////////////////////////////////
///////////////////////// urceni parametru modelu  /////////////////////////
////////////////////////////////////////////////////////////////////////////

  110: // Init
    runningBlockCase := 111; // prechod do dalsiho CASE

  111: // Main
    fb_parametrySOTD(omega_1 := omega1, omega_2 := omega2, K_s := K, bod_1.re
:= G1.re,  bod_1.im := G1.im, bod_2.re := G2.re,  bod_2.im := G2.im, a_2=>sotdA2,
a_1=>sotdA1, T_d=>sotdTau, K_p=>sotdK); // vypocet parameru modelu
    runningBlockCase := 120; // prechod do dalsiho CASE
    completed := TRUE; // priznak ze je identifikace dokoncena

////////////////////////////////////////////////////////////////////////////
////////////////////// priprava na dalsi identifikaci //////////////////////
////////////////////////////////////////////////////////////////////////////

  120: // Init
    runningBlockCase := 121; // prechod do dalsiho CASE

  121: // Main
    webOutputString  :='Identifikace  soustavy  byla  dokončena';  //  hlaseni
uzivateli
    runningBlockCase := 00; // prechod do dalsiho CASE
    buttonIsPressed := FALSE; // uvolneni tlacitka

  END_CASE;

END_FUNCTION_BLOCK
```