



APLIKACE PRO VÝPOČET SEŘÍZENÍ PID REGULÁTORŮ PRO ZAŘÍZENÍ S OS ANDROID

Diplomová práce

Studijní program:

N2301 Strojní inženýrství

Studijní obor:

Výrobní systémy a procesy

Autor práce:

Bc. Jiří Jelínek

Vedoucí práce:

Ing. Michal Moučka, Ph.D.

Katedra výrobních systémů a automatizace





Zadání diplomové práce

APLIKACE PRO VÝPOČET SEŘÍZENÍ PID REGULÁTORŮ PRO ZAŘÍZENÍ S OS ANDROID

Jméno a příjmení: **Bc. Jiří Jelínek**
Osobní číslo: **S19000367**
Studijní program: **N2301 Strojní inženýrství**
Studijní obor: **Výrobní systémy a procesy**
Zadávací katedra: **Katedra výrobních systémů a automatizace**
Akademický rok: **2020/2021**

Zásady pro vypracování:

1. Seznamte s principem vývoje aplikací pro operační systém Android.
2. Navrhňte strukturu aplikace pro výpočet seřízení parametrů PID regulátorů.
3. Vyberte výpočetní algoritmy parametrů regulátorů. Výběr konzultujte s vedoucím diplomové práce.
4. Ve vhodném vývojovém prostředí (AndroidStudio, VisualStudio) aplikaci naimplementujte.
5. Ověřte funkčnost a správnost aplikace.

Rozsah grafických prací:
Rozsah pracovní zprávy:
Forma zpracování práce:
Jazyk práce:

dle potřeby
cca 45 stran
tištěná/elektronická
Čeština



Seznam odborné literatury:

- [1] BALÁTĚ M. Automatické řízení. Praha: BEN – technická literatura, 2003. ISBN 978-80-7300-020-2.
- [2] DARWIN I. F. Java kuchařka programátora. Brno: Computer Press, 2006. ISBN 978-80-251-0944-5.
- [3] DWYER A. O. handbook of PID and PID controller tuning rules (third edition). London: Imperial College Press, 2009. ISBN 978-1848162426.
- [4] HOFREITER M. Základy automatického řízení (skriptum). Praha: České vysoké učení v Praze, 2012. ISBN 978-80-7372-297-5.
- [5] LACKO L. Vývoj aplikací pro Android. Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.
- [6] SHILDT H. Java 7 – Výukový kurz. Brno: Computer Press, 2013. ISBN 978-80-251-3748-2.

Vedoucí práce:

Ing. Michal Moučka, Ph.D.
Katedra výrobních systémů a automatizace

Datum zadání práce:

19. listopadu 2020

Předpokládaný termín odevzdání:

19. května 2022

prof. Dr. Ing. Petr Lenfeld
děkan

L.S.

Ing. Petr Zelený, Ph.D.
vedoucí katedry

V Liberci dne 5. března 2020

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

31. května 2021

Bc. Jiří Jelínek

PODĚKOVÁNÍ

Především bych chtěl vyjádřit své díky vedoucímu této práce Ing. Michalu Moučkovi, Ph.D. za to, že vymyslel, a pomáhal mi realizovat toto jednak složité, pracné, frustrující, ale také důstojné, a hodnotné téma diplomové práce.

Tato diplomová práce mi umožnila konečně nabýt přesvědčení, že dokáži samostatně řešit komplexní technické problémy. Navíc jsem se dozvěděl hodně o hodnotě sebekázně, a soustavné tvůrčí práci.

CZ: Tato práce byla [částečně] podpořena Studentskou grantovou soutěží Technické univerzity v Liberci v rámci projektu Optimalizace v oblasti výrobních systémů, 3D technologií a automatizace č. SGS-2019-5011

EN: This work was [partly] supported by the Student Grant Competition of the Technical University of Liberec under the project Optimization of manufacturing systems, 3D technologies and automation No. SGS-2019-5011

TÉMA: APLIKACE PRO VÝPOČET SEŘÍZENÍ PID REGULÁTORŮ PRO ZAŘÍZENÍ S OS ANDROID

ANOTACE: Předmětem diplomové práce je vytvoření aplikace pro operační systém Android. Úkolem aplikace je tvorba (identifikace) matematického modelu dynamického systému, na základě naměřených, nebo simulovaných externích dat, a využit takto získaný model k vhodnému seřízení parametrů PID regulátoru. Implementovaný výpočtový aparát využívá – minimalizace hodnoty integrační odchylky, metody optimalizace Flexibilního Simplexu (metoda Nelder–Mead) a metody numerické integrace Runge–Kutta čtvrtého řádu (metoda RK4).

KLÍČOVÁ SLOVA: mobilní aplikace, Android OS, identifikace dynamických systémů, PID Regulace, minimalizace odchylky integrace, metoda RK4, metoda Nelder–Mead, MATLAB

THEME: APPLICATION FOR CALCULATION OF PARAMETERS OF PID CONTROLLER FOR DEVICES WITH OS ANDROID

ANNOTATION: Subject of this thesis is creation of application for operating system Android. Purpose of the application is to create (identify) mathematical model of dynamic system, based on measured or simulated external data, and use this model in proper tuning of parameters of PID controller. Implemented computational apparatus is using- minimalization of integral absolute error, numerical optimisation method of downhill simplex (method Nelder–Mead) and numerical integration method of Runge–Kutta fourth order (method RK4).

KEYWORDS: mobile application, Android OS, dynamic systems identification, PID regulation, integral absolute error, method RK4, method Nelder-Mead, MATLAB

Zpracovatel: TU v Liberci, Fakulta strojní, Katedra výrobních systémů a automatizace

Počet stran : 108
Počet příloh : 1
Počet obrázků : 129
Počet tabulek : 23
Počet modelů nebo jiných příloh: 0

OBSAH

ÚVOD	9
1 REŠERŠE.....	10
1.1 VÝSLEDEK REŠERŠE.....	17
2 APLIKACE.....	18
2.1 OBECNÁ STRUKTURA APLIKACE.....	18
2.2 NAVIGACE APLIKACÍ	19
2.3 AKTIVITA DATA.....	21
2.4 AKTIVITA IDENTIFIKACE.....	23
2.5 AKTIVITA ŘÍZENÍ.....	27
3 VÝPOČTOVÝ PRINCIP.....	31
3.1 DYNAMICKÝ SYSTÉM	31
3.2 IDENTIFIKACE	33
3.2.1 Identifikace systémů obecně	33
3.2.2 Implementace v Aktivitě identifikace	34
3.2.3 Detailní postup při identifikaci.....	36
3.3 ŘÍZENÍ.....	39
3.3.1 Princip PID regulátoru	40
3.3.2 Princip seřizování regulačního obvodu aplikací.....	42
3.3.3 Detailní postup při seřizování PID regulátoru.....	44
3.4 OPTIMALIZAČNÍ METODA FLEXIBILNÍHO SIMPLEXU	48
3.4.1 Algoritmus metody NM	49
3.4.2 Ukončovací podmínky	51
3.5 INTEGRAČNÍ METODA RK4.....	52
3.5.1 RK4	52
3.5.2 Převod ODR vyššího řádu na soustavu ODR prvního řádu	52
3.5.3 Implementace metody RK4.....	53
4 VERIFIKACE	55
4.1 VERIFIKACE INTEGRACE (METODA RK4)	55

4.2	VERIFIKACE IDENTIFIKACE (METODA NELDER-MEAD)	63
4.2.1	Generování verifikačních signálů	63
4.2.2	Vyhodnocení identifikace verifikačních signálů	65
4.2.3	Shmutí verifikace identifikace:	90
4.3	VERIFIKACE SEŘIZOVÁNÍ PID REGULÁTORU	91
4.3.1	Shmutí verifikace seřizování PID regulátoru:	106
ZÁVĚR		107
POUŽITÁ LITERATŮRA A ZDROJE:		108

ÚVOD

Tato diplomová práce se zabývala tvorbou inženýrské mobilní aplikace pro zařízení s OS Android. Úkolem této aplikace bylo za pomoci naměřených, popřípadě simulovaných hodnot, odezvy dynamického systému na vstupní signál tento dynamický systém identifikovat. A následně pomocí takto identifikovaného modelu systému vhodně seřadit parametry PID regulátoru pro daný dynamický systém.

Lze se říci, že šlo o snahu vytvořit zjednodušenou mobilní alternativu k MATLAB: System Identification Toolbox a MATLAB: PID Tuner, jež lze považovat za hlavní inspiraci této práce.

Aplikace má sloužit všem, kteří se zabývají řízením dynamických systémů, ať už v technické praxi, v rámci studia, či pouhého laického zajmu, kdy nebudou potřebovat, k identifikování a řízení dynamického systému nic jiného než své mobilní zařízení.

Cílem práce bylo vyvinout praktický software pro OS Android, který bude schopen plnit následující úlohy:

- A. Bude schopen načíst externí data měření, uložená v mobilním zařízení.
- B. Bude schopen data interně ukládat a spravovat
- C. Bude schopen data aproximovat modelem dynamického systému n -tého řádu
- D. Bude schopen dle modelu dat optimálně zvolit parametry PID regulátoru
- E. Bude uživatelsky přívětivý

Řešení úkolu se skládalo z několika úrovní:

Nejprve bylo nutno zvolit správný výpočtový aparát, který je efektivní, odolný, relativně jednoduchý na implementaci a který poskytuje uspokojivé výsledky.

Následoval návrh struktury aplikace, tak aby vytvářela jeden koherentní celek s ohledem na všechny funkční prvky a mechaniky aplikace.

V neposlední řadě bylo nutno celou aplikaci naimplementovat v OS Android. Výsledná implementace, měla být srozumitelná, uživatelsky přívětivá, stabilní a poskytovat validní výsledky. Jež bylo nutno následně ověřit pomocí výpočtového softwaru MATLAB.

1 REŠERŠE

Vzestup mobilních zařízení je již několik let neoddiskutovatelným faktem. Jsme jimi obklopeni prakticky neustále, vždy a všude, přičemž neustálá poptávka zákazníků, nutí výrobce uvádět na trh stále výkonnější a výkonnější zařízení, za stále nižší a nižší ceny.

Žádný hardware nemůže existovat a být úspěšný bez solidní softwarové základny. Dá se říci, že software na tato zařízení je šířen výhradně pomocí takzvaných „app storů“. Jedná se o často vestavěnou funkcionalitu těchto zařízení, kdy provozovatel operačního systému, distribuuje software třetích stran na své zařízení za podíl na zisku z jeho prodeje. V současné době jsou nejdůležitějšími představiteli App Store, který distribuuje aplikace na zařízení s operačním systémem iOS, a Google Play, jež poskytuje stejnou službu pro zařízení s operačním systémem Android. Dalším zástupcem je např. AppGallery, od čínské společnosti Huawei.

Na začátku roku 2020 bylo na Google Play k dispozici okolo 3 miliónů aplikací. Ani jedna z nich však nesplňovala představy praktického inženýrského nástroje, který by byl schopen plně využít potenciálu tohoto hardwaru na poli inženýrské kybernetiky.

Rešerše zkoumala možnosti aplikací, jež se do této doby zabývaly problematikou automatického řízení. Všechny jsou dostupné k veřejnému stažení skrze službu Google Play.

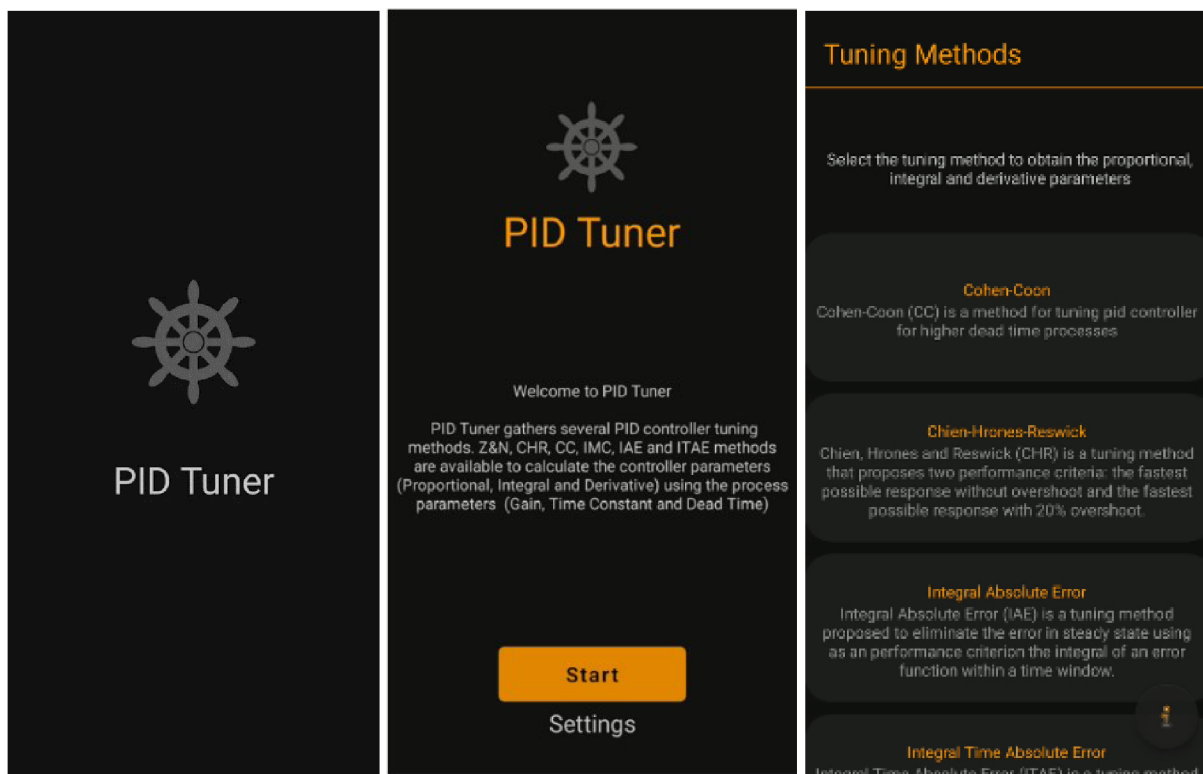
a) PID Tuner

Aplikace PID Tuner představuje vcelku vydařenou výukovou pomůcku, která umožňuje výpočet parametrů PID regulátoru, při zadání parametrů dynamického systému prvního řádu. Kterými jsou: statické zesílení, časová konstanta a zpoždění systému¹.

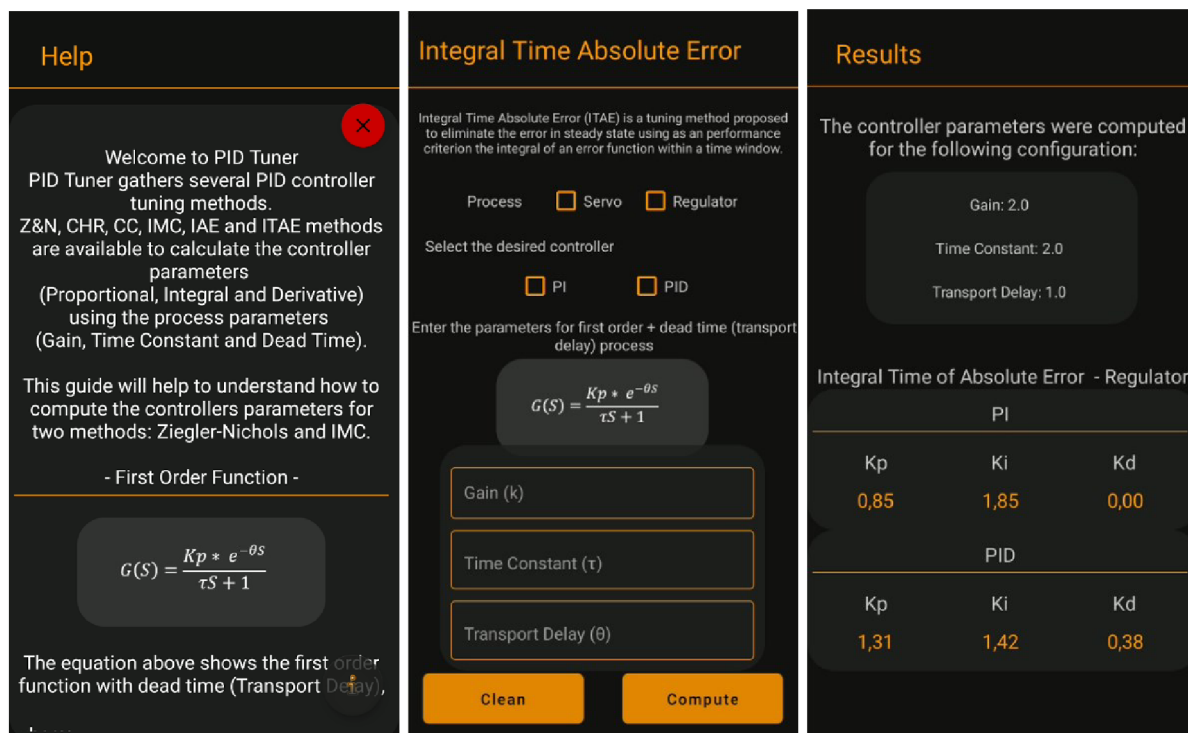
Jak je vidět na obr. 1.1, aplikace nabízí řadu metod seřízení regulátoru, jako jsou: Ziegler-Nichols, integrál absolutní regulační odchylky, časový integrál absolutní odchylky a několik dalších. Tato aplikace nám umožňuje spočítat všechny tyto metody pro jeden systém a následně porovnat výsledky. Potenciál všech těchto metod je však marněn tím, že jsou implementovány pouze na systém prvního řádu, což podkopává význam implementace některých metod (obr. 1.2). Nicméně jak uvádí informace, jež je k dispozici v aplikaci samotné (obr. 1.2), aplikace má složit pouze jako výuková pomůcka.

¹ Pro více informací o dynamických systémech viz. 3.2 Identifikace

Zhodnocení: Jedná se o funkční demonstrační, studijní, vizuálně hezky zvládnutou pomůcku, nicméně možnosti výpočtu v ní implementované jsou velice slabé a nevhodné pro cokoliv jiného než demonstraci základních principů, automatického řízení.



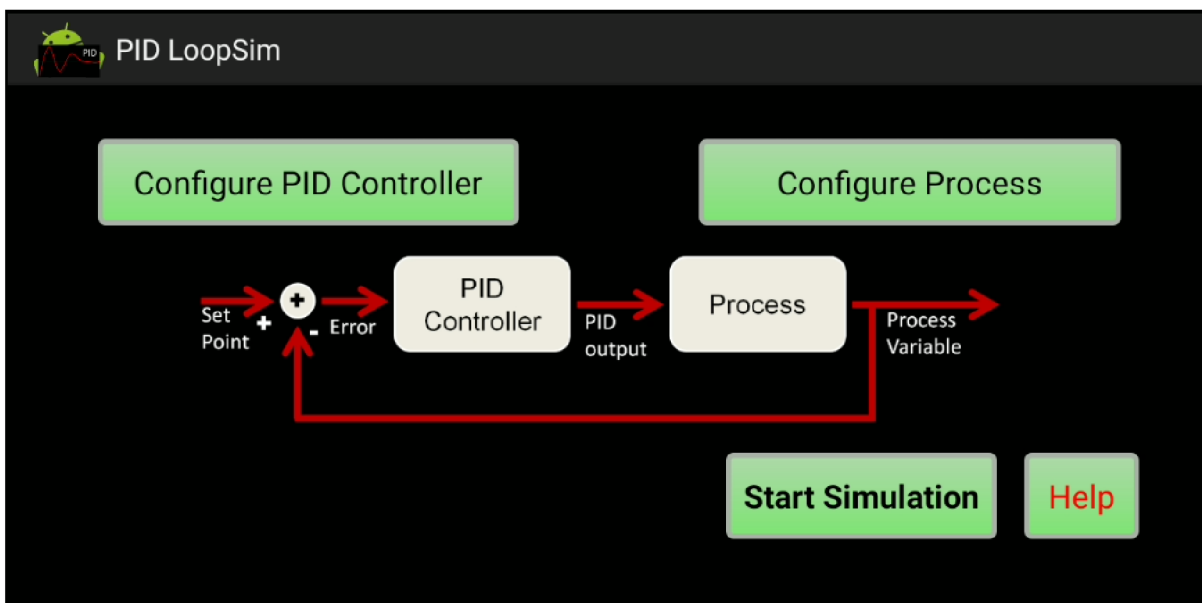
Obr. 1.1 PID Tuner nabídka a navigace aplikací



Obr. 1.2 PID Tuner výpočetní část

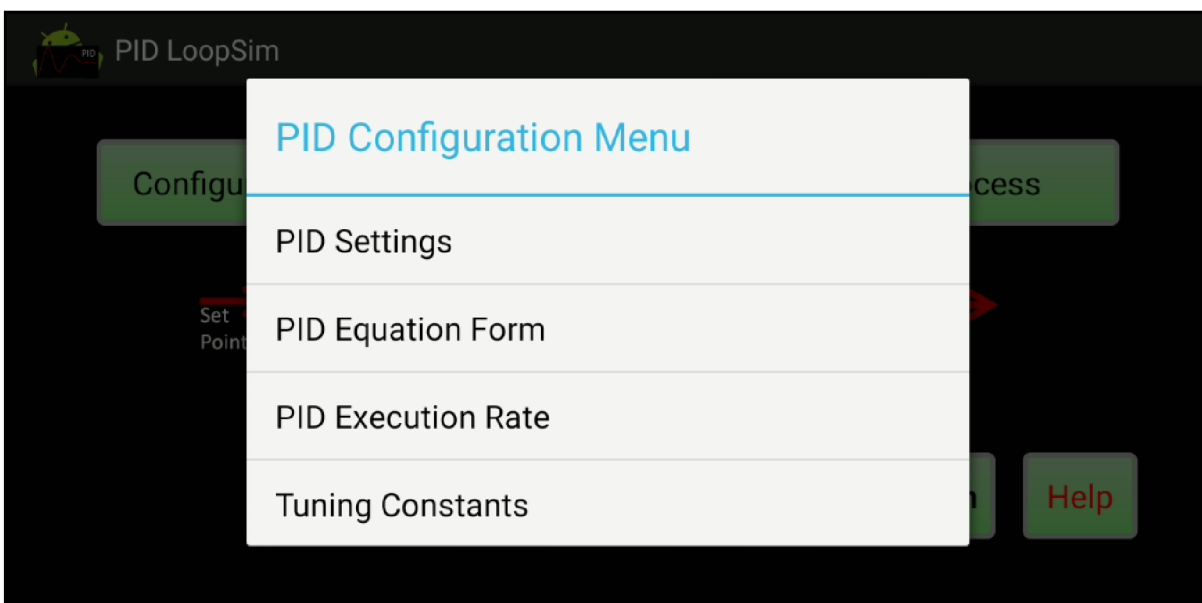
b) PID LoopSim

PID LoopSim je rovněž simulační výpočtovou aplikací. Tato aplikace se však snaží o grafickou vizualizaci simulace regulace regulační smyčky dynamického systému pomocí PID regulátoru.

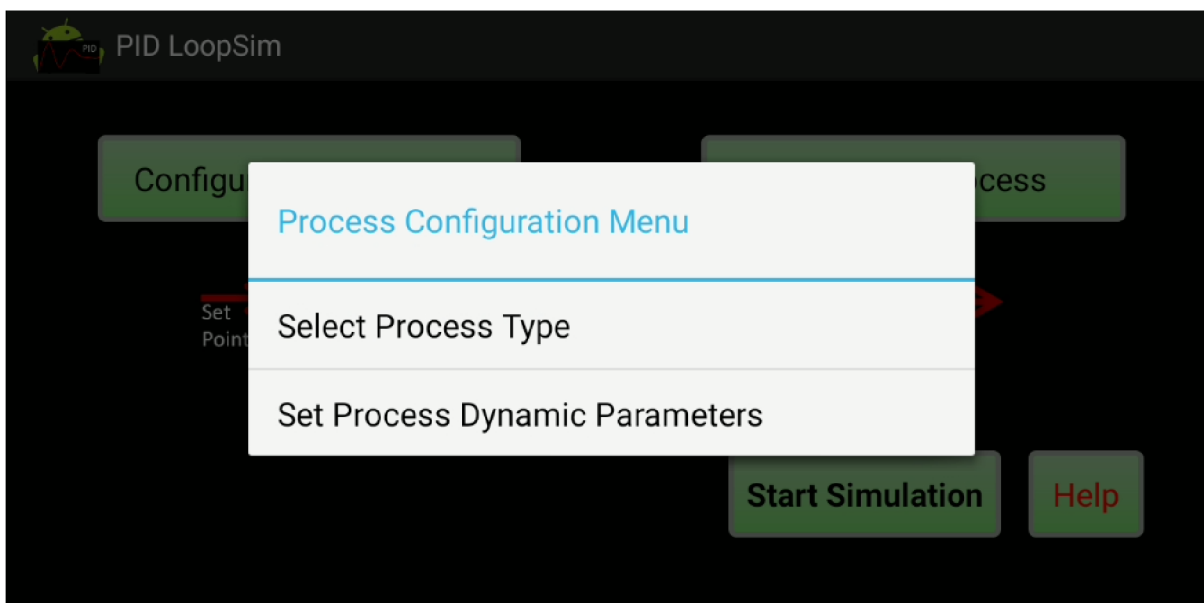


Obr. 1.3 PID LoopSim Hlavní nabídka

Na první pohled se může z množství nastavení zdát, že tato aplikace, má potenciál pro reálné použití v praktickém životě díky svému obsáhlému menu (obr. 1.4).

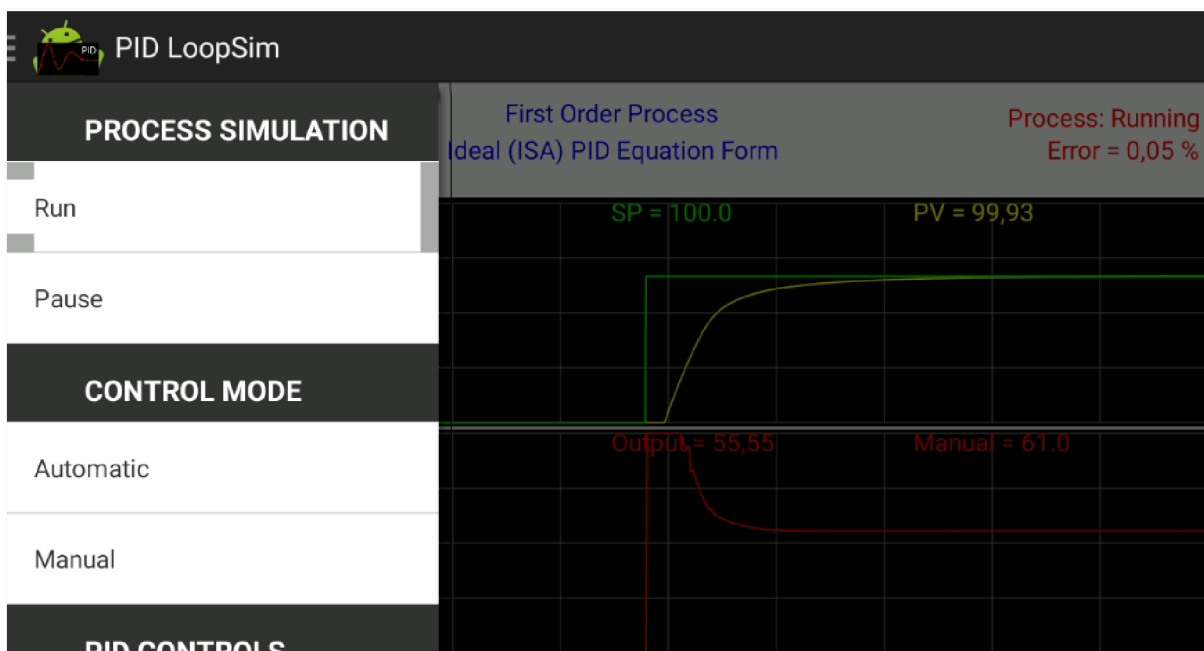


Obr. 1.4 PID LoopSim nastavení regulátoru



Obr. 1.5 PID LoopSim nastavení procesu

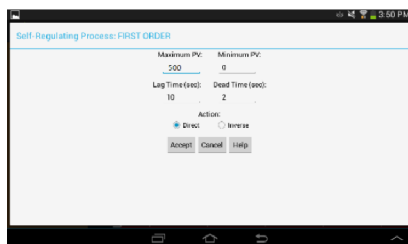
Nicméně při prostudování dostupné dokumentace zjistíme, že se jedná o relativně jednoduchý simulátor prvního a druhého řádu (v případě placeného rozšíření) dynamického systému (obr. 1.7, obr. 1.8). Je jasné, že i zde jsou možnosti seřizování velice omezeny, přestože grafická vizualizace výsledků je na vyšší úrovni. Naprosto chybí automatické seřízení parametrů PID regulátoru. A k simulaci řízení pomocí PID je nutno znát koeficienty procesního modelu² dynamického systému (obr. 1.7, Obr. 1.8).



Obr. 1.6 PID LoopSim Simulace procesu

² Pro více informací, o procesních modelech systémů, viz. 3.2 Identifikace

A- First Regulating Process - 1st order:

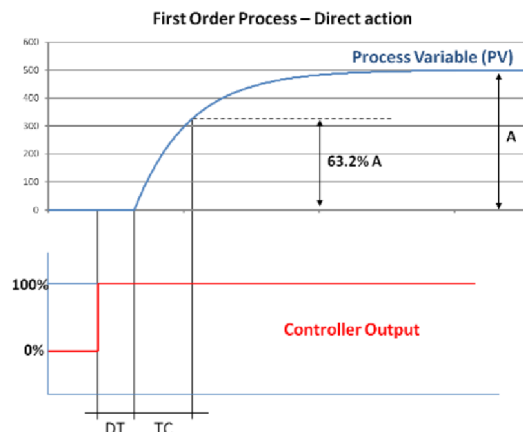


Maximum PV: Steady state value of the process variable (in engineering units) when the PID controller output is 100%.

Minimum PV: Steady state value of the process variable (in engineering units) when the PID controller output is 0%.

Lag Time (TC): Time constant of the process. 63.2% of process step response (see graphic below).

Dead Time (DT): Process delay (see graphic below).



Obr. 1.7 PID LoopSim systém prvního řádu Zdroj: PID LoopSim. DroidBus/TCP [online]. Cubac Apps ©2013 [cit. 18.3.2021].

Dostupné z: <http://www.droidbus.com/PIDLoopSim/#help>

B- First Regulating Process - 2nd order:

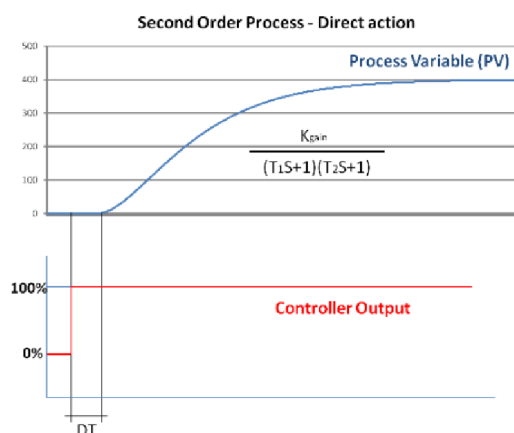


Maximum PV: Steady state value of the process variable (in engineering units) when the PID controller output is 100%.

Minimum PV: Steady state value of the process variable (in engineering units) when the PID controller output is 0%.

Lag Time 1 / Lag Time 2 (T1 and T2): Time constants of the process (see graphic below).

Dead Time (DT): Process delay (see graphic below).



Obr. 1.8 PID LoopSim systém druhého řádu Zdroj: PID LoopSim. DroidBus/TCP [online]. Cubac Apps ©2013 [cit. 18.3.2021].

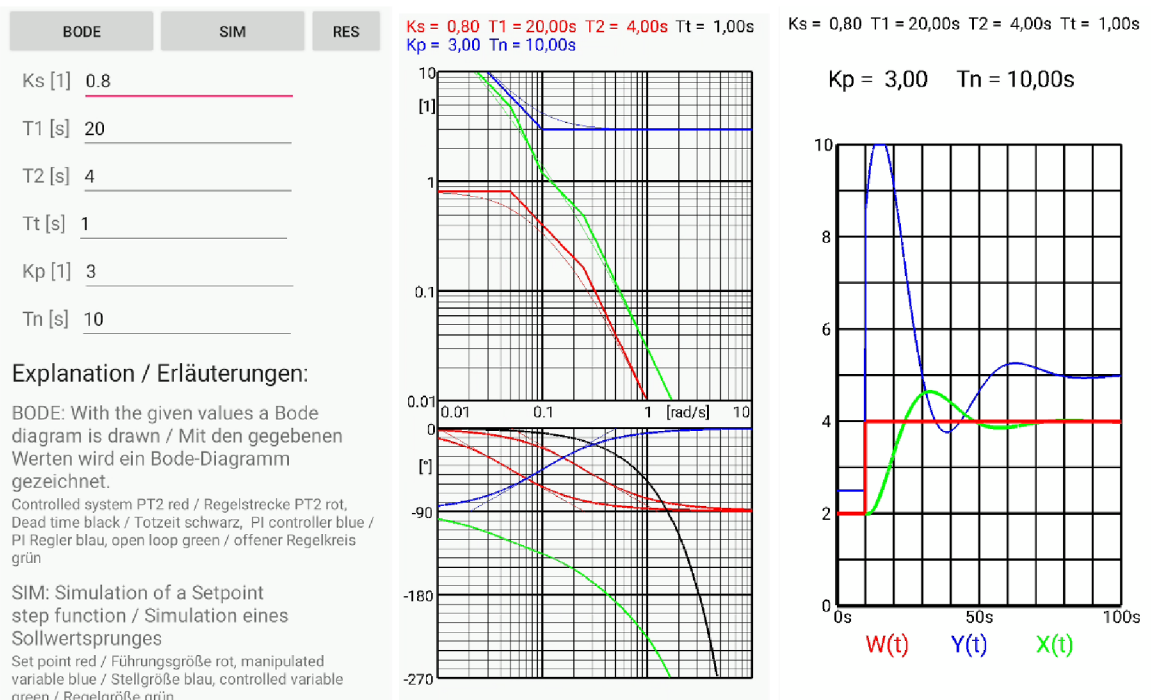
Dostupné z: <http://www.droidbus.com/PIDLoopSim/#help>

Zhodnocení: Jedná se o nástroj výpočtu regulační smyčky³ s množstvím nastavení. Nicméně, je nutné dopředu znát identifikaci systému. Vizualizace vývoje odezvy systému na regulaci je zajímavý, ale pro praktické řízení nedůležitý prvek (zajímá nás pouze časový horizont ustálení na požadovanou hodnotu). Aplikace je tedy pro praxi rovněž nevhodná. Uživatelské provedení má řadu grafických a funkčních nedostatků.

c) Control loop

Aplikace Control loop je jednoduchá aplikace, pro simulaci seřízení PI regulátoru. Seřízení probíhá pomocí obrazového přenosu otevřené smyčky. Pro svou funkci vyžaduje koeficienty procesního modelu řízeného systému druhého řádu. Kromě diagramu samotného regulace umožňuje vykreslení Bodeho diagramu frekvenční charakteristiky.

³ Pro více informací o regulačních smyčkách viz. 3.3 Regulace



Obr. 1.9 control loop ukázka celé aplikace

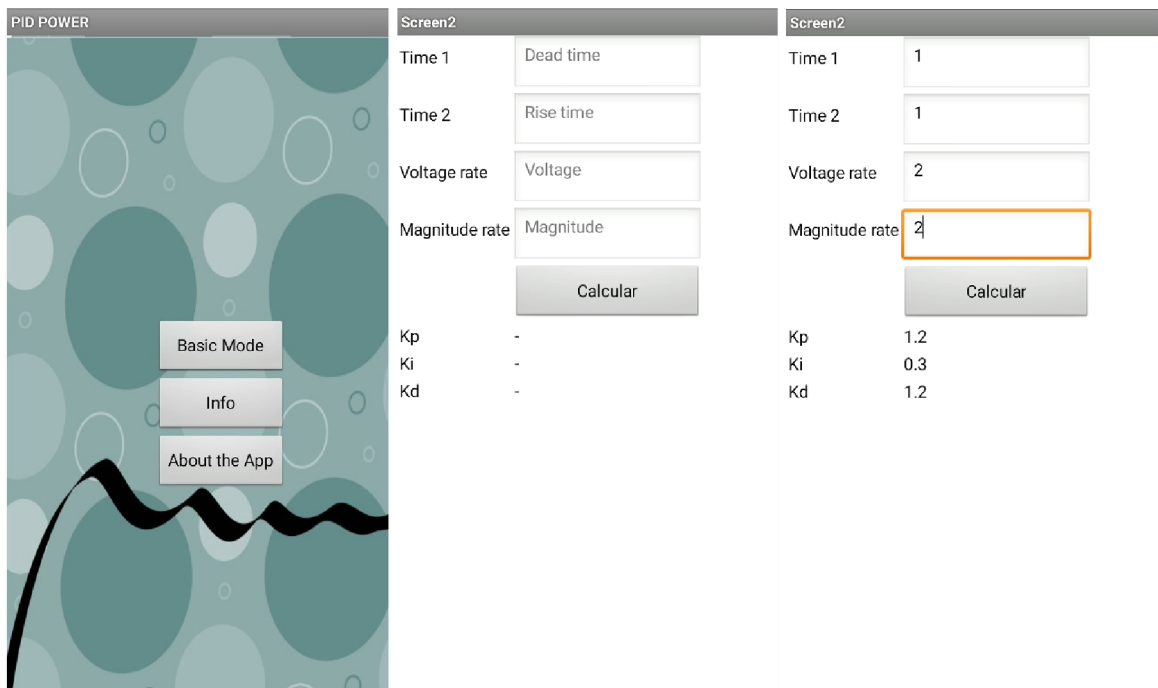
Zhodnocení: Aplikace toho nenabízí mnoho. Seřízení PI regulace probíhá pouze v otevřené smyčce, na omezeném modelu systému a není poskytnuta možnost automatického seřízení regulátoru. Jediným plusem je jednoduchost celé aplikace a prezentace výsledků. Pro technickou praxi naprosto nevhodné.

d) PID POWER

Nedokončený projekt, který v sobě má implementovaný výpočet minimalizace koeficientů regulátoru dle procesního modelu 2. řádu. Je nutno doplnit informace o systému ($T1$ – Time 1, $T2$ – Time 2, K_c – Magnitude rate, a frekvenci vzorkování f - Voltage rate)⁴.

Zhodnocení: Praktické využití je minimální, obr. 1.10 postihuje celou aplikaci.

⁴ Pro více informací o procesních modelech a jejich vstupních proměnných viz. 3.2 Identifikace

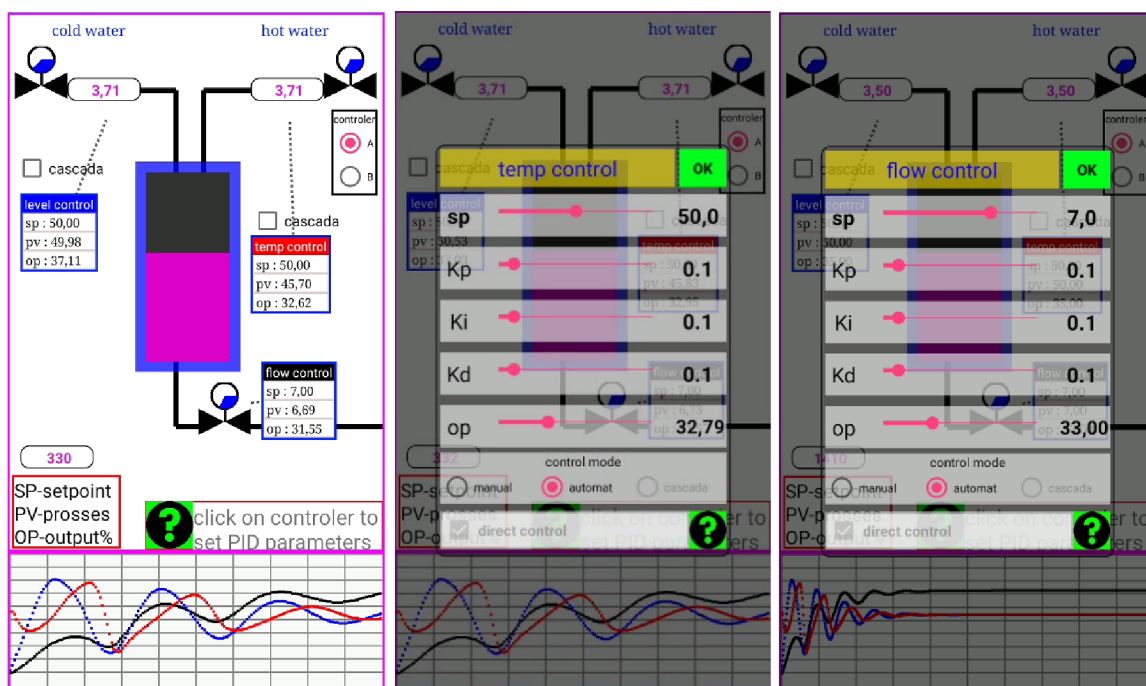


Obr. 1.10 PID POWER

e) PID control simulation

Simulační software, jenž simuluje směšování teplé a studené vody, technicky propracovanější aplikace (animace a chování simulovaného systému, v reálném čase), nicméně se nedá využít pro nic jiného než výuku, popřípadě simulaci velice primitivního systému směšování teplé a studené vody.

Zhodnocení: Celá aplikace je na systémové úrovni propracovaná, nicméně praktické využití je zde opět velice omezené na pouhou výukovou pomůcku.



Obr. 1.11 PID control simulation

Další alternativy

Další alternativu představuje aplikace MATLAB Mobile, kde může uživatel pracovat s MATLAB skripty uloženými na MATLAB Drive. V těchto skriptech je možné vytvořit si skript s fungujícím modelem minimalizační funkce a skript pro seřízení PID regulátoru a následně zpracovávat pomocí těchto skriptů data uložená na MATLAB Drive. Toto řešení je tedy podmíněno vlastnictvím licence softwaru MATLAB. Nicméně zde stále chybí podpora dedikovaných modulů pro identifikaci a PID regulaci (MATLAB: System Identification Toolbox, a MATLAB: PID Tuner) v samotném mobilním zařízení. Práce se skripty je v tomto případě těžkopádná. Také chybí možnost pracovat s daty přímo uloženými v mobilním zařízení (vše musí být uloženo na MATLAB Drive). Teprve poté je možno pracovat s daty na mobilním zařízení. Jelikož je však MATLAB Mobile primárně výpočtový software a k výpočtu identifikace systémů a jejich řízení ho lze pouze využít až po vložení příslušného uživatelského kódu, není plnohodnotně zahrnut v rešerši.

1.1 VÝSLEDEK REŠERŠE

Jak lze vidět, žádná z doposud dostupných aplikací nespĺňuje požadavky, vytyčené touto prací. Konkrétně: A. schopnost pracovat s naměřenými daty, B. schopnost Identifikovat model dynamického systému n-tého řádu, C. schopnost spolehlivě seřídít PID regulátor podle modelu dynamického systému. Některé z aplikací do jisté míry plní bod C. nicméně i zde jsou značná omezení.

Provedení aplikací značně kolísá. Od relativně profesionálních, až po naprosto nedokončené amatérské projekty. Pro skutečné využití v technické praxi se však nehodí žádná z nich.

Vzhledem k tomu, že provedení rešeršovaných aplikací je značně odlišného formátu bylo finální srovnání vyjádřeno tabulkou tab. 1.1. Posuzované znaky byly primárně kvalitativního rázu:

Stupnice: 1 – vynikající, 2 – nadprůměrné, 3 – průměrné, 4 – dostačující, 5 – nedostačující

Název Aplikace	Provedení aplikace	Výpočetní aparát aplikace	Uživatelské prostředí aplikace	Praktické využití aplikace
PID Tuner	3	3	1	5
PID LoopSim	2	3	3	5
Control loop	3	3	4	5
PID POWER	4	4	4	5
PID control simulation	1	3	2	5

Tab. 1.1 srovnání rešeršovaných aplikací

Výsledek rešerše představoval primární impuls pro vytvoření této práce, která byla především reakcí na neexistenci vhodného softwaru pro praxi automatického řízení, na dané platformě.

2 APLIKACE

Aplikace byla vytvořena ve freewarovém IDE (Integrated Development Environment) Android Studio. Samotný kód je psán v programovacím jazyku JAVA, uzpůsobeném pro potřeby aplikací využívajících OS Android. Je navržena s co největším ohledem na modularitu, přičemž většina metod je napsána jako nezávislé knihovny spolu s co největším množstvím obecného kódu.

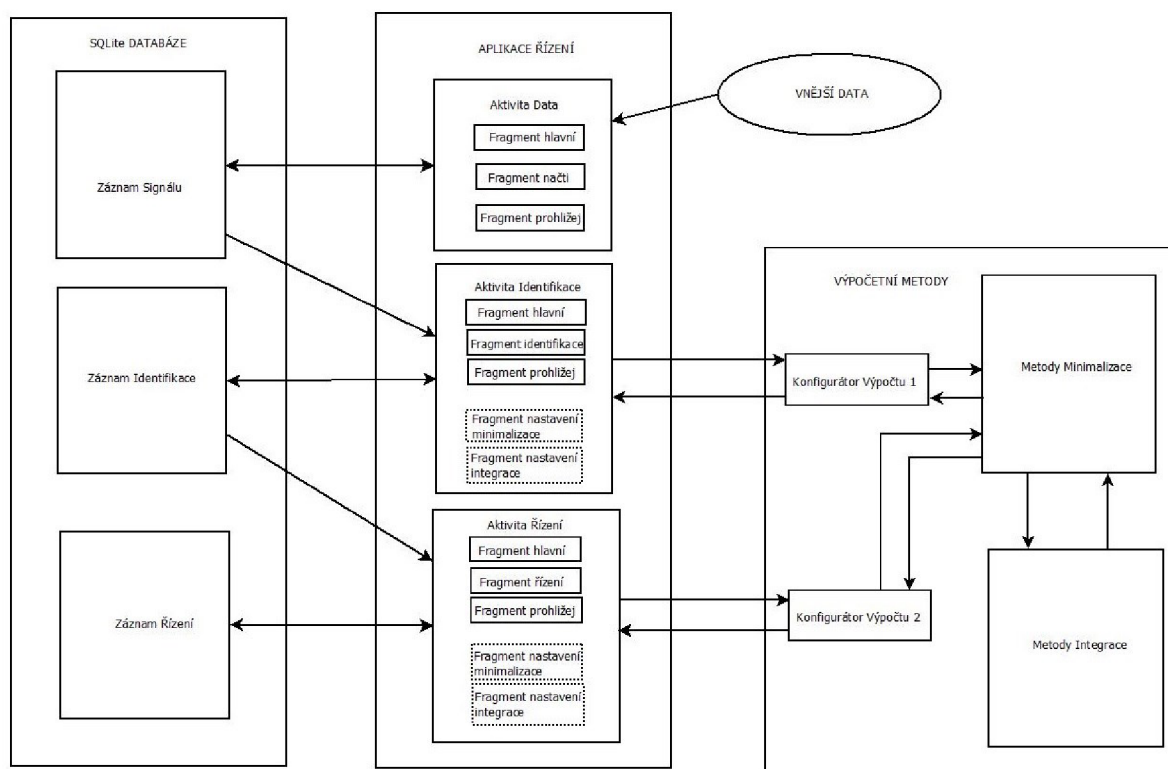
Pro ukládání informací o výpočtech je využíváno vestavěného nástroje OS Android: databáze SQLite.

2.1 OBECNÁ STRUKTURA APLIKACE

Aplikace je z uživatelského i vývojového hlediska rozdělena do tří hlavních celků (aktivit):

1. Aktivita pro načtení a předzpracování naměřených dat
2. Aktivita pro identifikaci parametrů systému
3. Aktivita pro seřízení parametrů PID regulátoru

Aktivita se dále skládají z menších celků (fragmentů), jež jsou zodpovědné za samotnou interakci s uživatelem na nejnižší úrovni.



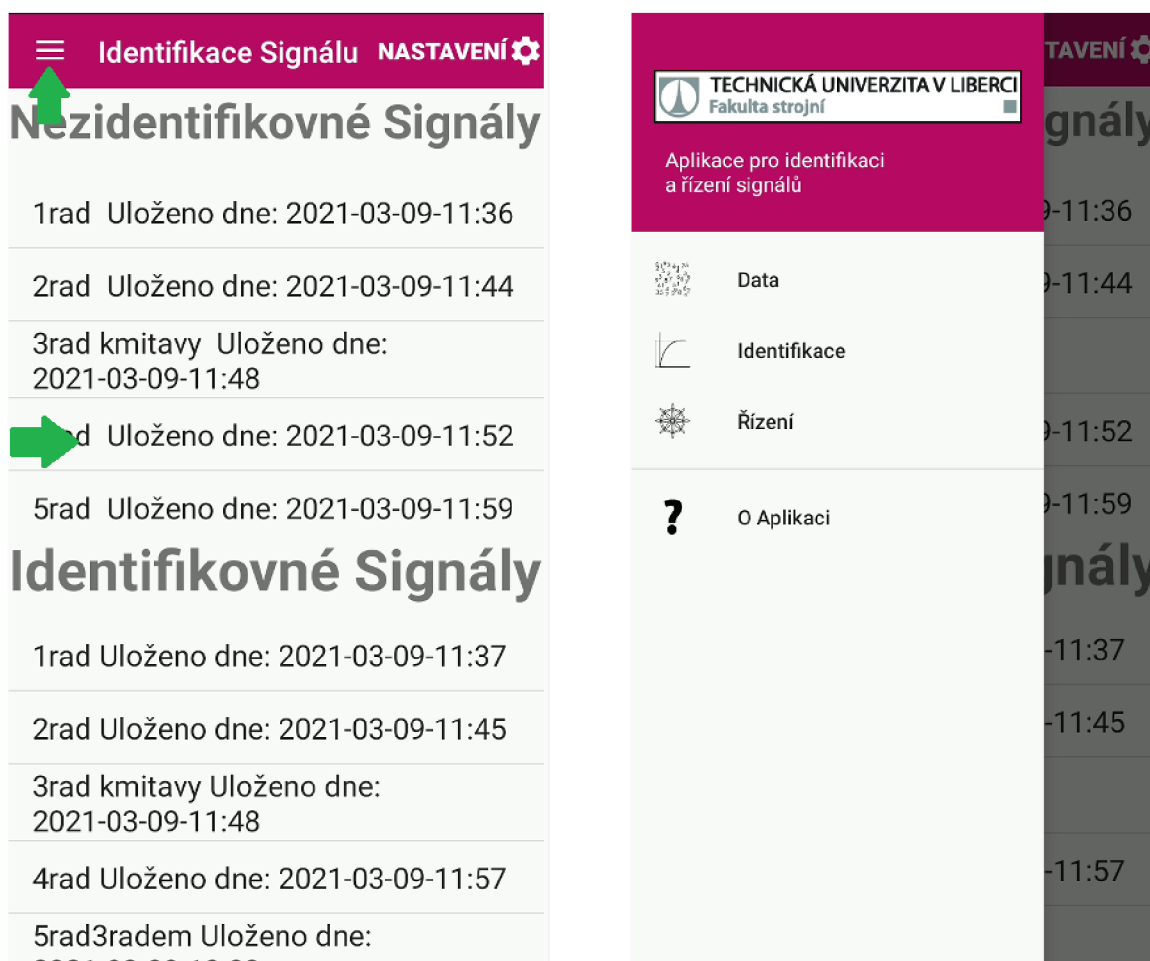
Obr. 2.1 Diagram obecné struktury aplikace

Jednotlivé aktivity jsou nezávislými entitami, jak ukazuje (obr. 2.1), a propojení mezi nimi je realizováno pouze pomocí SQLite databáze. Tímto způsobem je možno jednoduše a permanentně ukládat výsledky výpočtů a mezivýpočtů.

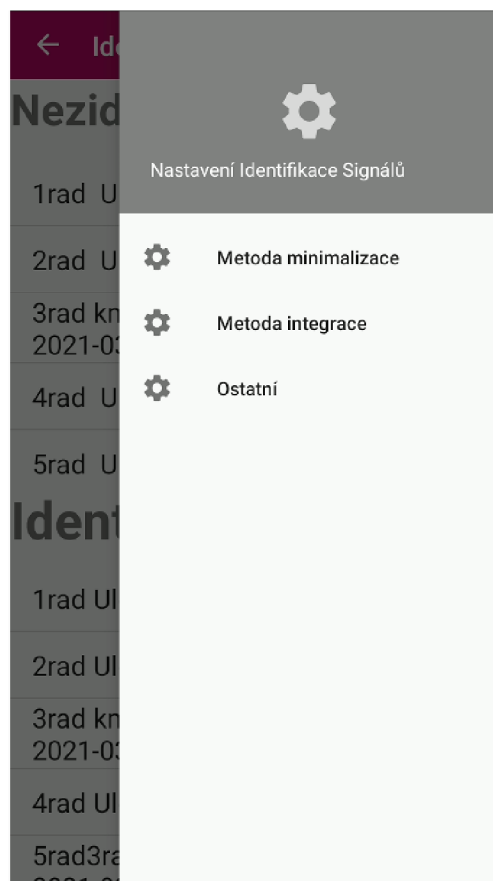
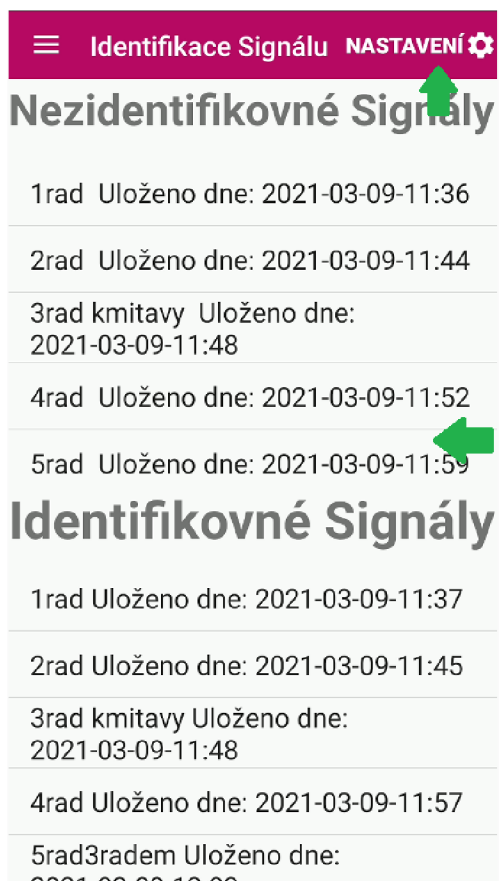
2.2 NAVIGACE APLIKACÍ

Navigace aplikací je realizována jednotným hlavním menu přístupným odkudkoliv v aplikaci tzv. „drawer menu“. Uživatel může volně vybírat mezi aktivitami (obr. 2.2). Toto menu se volá buďto pomocí ikony v levém horním rohu, nebo gestem „draw“ – „vytažením“, z levého okraje obrazovky. Pomocí stejné mechaniky je vytvořeno i menu nastavení (obr. 2.3), které se volá obdobně, avšak z pravé strany obrazovky. Při zpracování měřených dat uživatel zpravidla postupuje následovně: Aktivita Data - Aktivita Identifikace - Aktivita Řízení. Nicméně pro detailnější seřizování a identifikaci je zpravidla nutno se mezi těmito aktivitami volně pohybovat a „experimentovat“ s různými nastaveními identifikace a seřízením PID regulátoru.

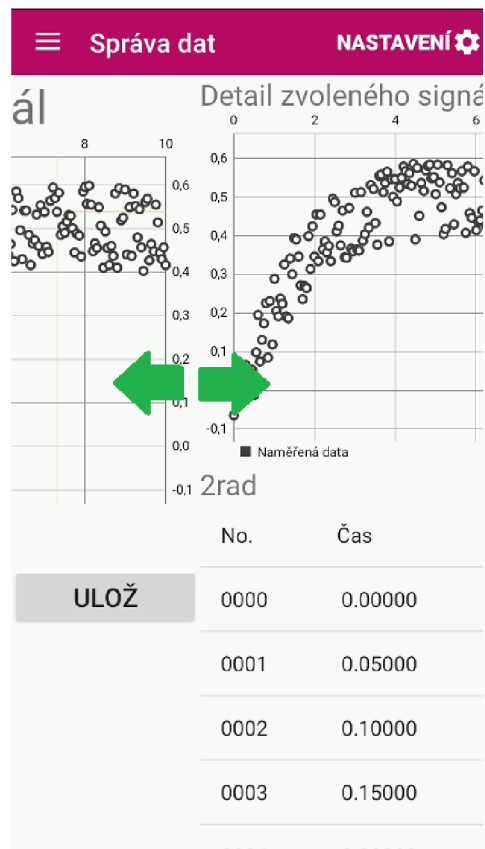
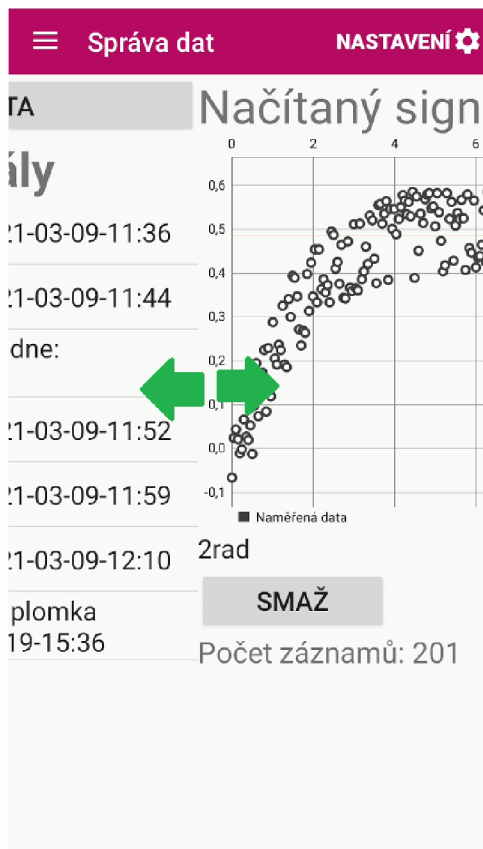
Jednotlivé aktivity dále obsahují fragmenty „panely“, sloužící pro interakci s uživatelem. Mezi nimi lze volně přecházet pomocí gesta „swipe“ – „přetažení“, ovládací prvek je oproti „drawer menu“ umístěn blíže středu obrazovky (obr.2.4).



Obr. 2.2 Hlavní menu „drawer“



Obr. 2.3 Nastavení menu „drawer“



Obr. 2.4 Navigace fragmenty

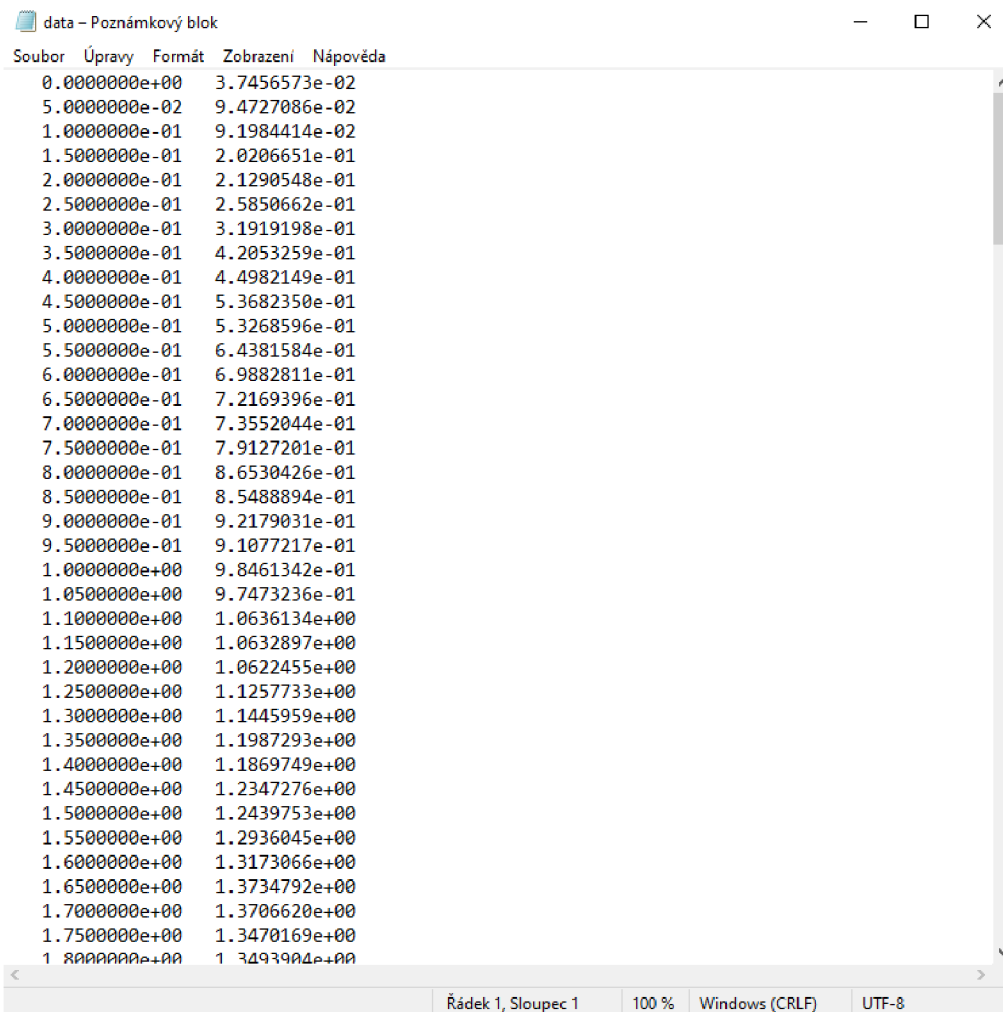
2.3 AKTIVITA DATA

Úkolem aktivity Data je načtení a předzpracování naměřených dat. V současnosti aplikace obsahuje pouze množnost vkládat externí data z měření jako textový dokument, ve formátu zobrazeném na (obr. 2.5). V prvním sloupci jsou hodnoty času měření a v druhém sloupci jsou hodnoty měření odezvy systému na jednotkový skok.

Dokument s takto naměřenými daty je uložen do paměti mobilního zařízení (interní nebo externí úložiště – SD karta). A následně je otevřen uvnitř aplikace, pomocí správce souborů (obr. 2.6).

Otevíraný soubor může mít v současné době pouze příponu .dat nebo .txt. Pokud je daný soubor správné přípony a formátu, aplikace si ho přečte a zobrazí jeho graf ve fragmentu Data (obr. 2.7).

Zde je možno načtený signál pojmenovat a uložit tlačítkem uložit do interní databáze aplikace. (obr. 2.8).



Soubor	Úpravy	Formát	Zobrazení	Nápvěda
0.000000e+00				3.7456573e-02
5.000000e-02				9.4727086e-02
1.000000e-01				9.1984414e-02
1.500000e-01				2.0206651e-01
2.000000e-01				2.1290548e-01
2.500000e-01				2.5850662e-01
3.000000e-01				3.1919198e-01
3.500000e-01				4.2053259e-01
4.000000e-01				4.4982149e-01
4.500000e-01				5.3682350e-01
5.000000e-01				5.3268596e-01
5.500000e-01				6.4381584e-01
6.000000e-01				6.9882811e-01
6.500000e-01				7.2169396e-01
7.000000e-01				7.3552044e-01
7.500000e-01				7.9127201e-01
8.000000e-01				8.6530426e-01
8.500000e-01				8.5488894e-01
9.000000e-01				9.2179031e-01
9.500000e-01				9.1077217e-01
1.000000e+00				9.8461342e-01
1.050000e+00				9.7473236e-01
1.100000e+00				1.0636134e+00
1.150000e+00				1.0632897e+00
1.200000e+00				1.0622455e+00
1.250000e+00				1.1257733e+00
1.300000e+00				1.1445959e+00
1.350000e+00				1.1987293e+00
1.400000e+00				1.1869749e+00
1.450000e+00				1.2347276e+00
1.500000e+00				1.2439753e+00
1.550000e+00				1.2936045e+00
1.600000e+00				1.3173066e+00
1.650000e+00				1.3734792e+00
1.700000e+00				1.3706620e+00
1.750000e+00				1.3470169e+00
1.800000e+00				1.3493904e+00

Obr. 2.5 Textový dokument obsahující externí naměřené hodnoty

☰ Správa dat NASTAVENÍ ⚙️

NAČTI DATA ↑

Uložené signály

1rad	Uloženo dne: 2021-03-09-11:36
2rad	Uloženo dne: 2021-03-09-11:44
3rad kmitavy	Uloženo dne: 2021-03-09-11:48
4rad	Uloženo dne: 2021-03-09-11:52
5rad	Uloženo dne: 2021-03-09-11:59
6rad	Uloženo dne: 2021-03-09-12:10

☰ 1 🔍 ⋮

A > DCIM > Vzoroky signálů > 2. řád > 1

📁 Velké soubory 🔄 Tento týden

SOUBORY VE SLOŽCE 1 ☰

📄

data.dat
6,83 kB 12. 8. 2...

📄

y"+3y'+2y=u...
110 B 12. 8. 20...

Obr. 2.6 Aktivita Data: načítání externích dat

☰ Správa dat NASTAVENÍ ⚙️

Načítaný signál

■ Naměřená data

Nový Signál obrázek diplomka

SMAŽ
ULOŽ

Počet záznamů: 201

☰ Správa dat

Načítaný signál

■ Naměřená data

Nový Signál obrázek2 diplomka

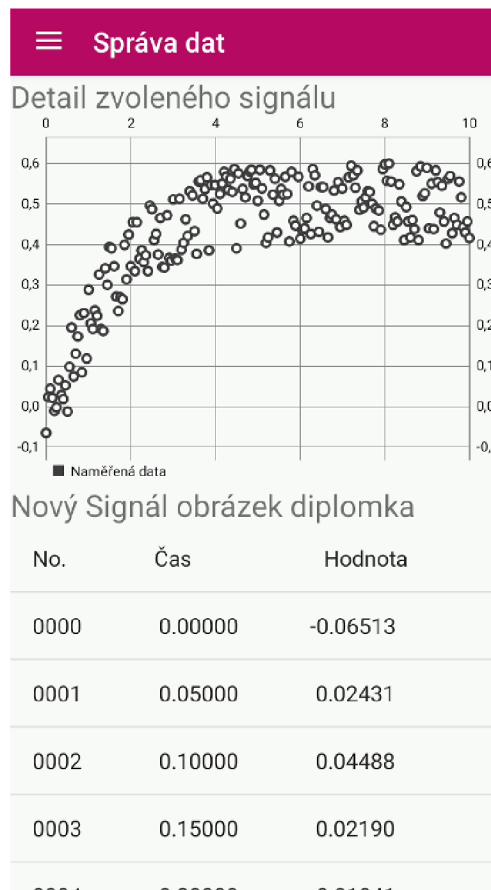
SMAŽ
ULOŽ

Počet záznamů: 801

Obr. 2.7 Data fragment Načti



Obr. 2.8 Data fragment Hlavní



Obr. 2.9 Data fragment Prohlížeč

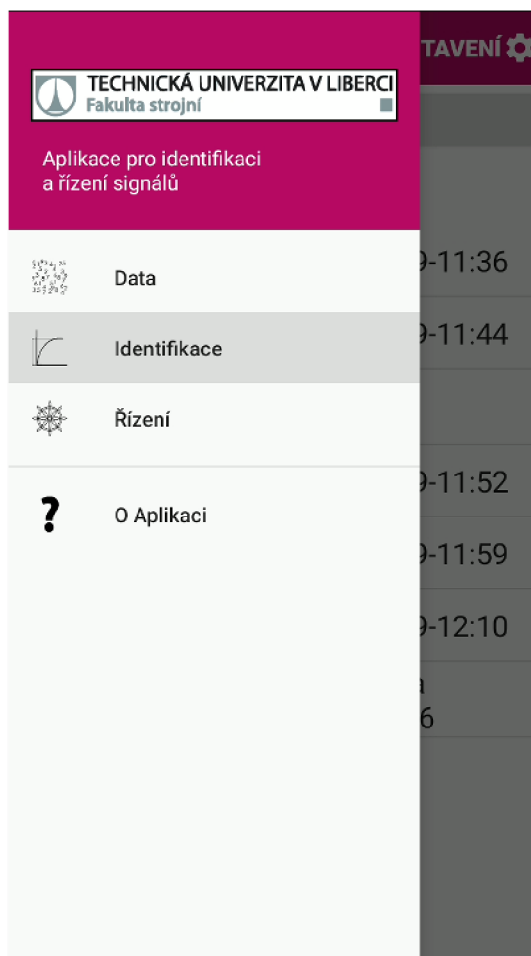
Pokud si uživatel přeje daný signál znovu zobrazit, provést na něm úpravy, či ho smazat lze ho vybrat z nabídky uložených signálů. Aktivita Data rovněž obsahuje fragment prohlížeč, který umožňuje detailní prohlížení hodnot (obr. 2.9).

Tyto fragmenty představují panely a jsou vzájemně propojeny, takže mezi nimi lze přecházet jednoduchým tažením do strany, tzv. „swipe“, pořadí fragmentů je: fragment Hlavní, fragment Načti, fragment Prohlížeč.

2.4 AKTIVITA IDENTIFIKACE

Úkolem aktivity identifikace je vytvořit matematický model dynamického systému, z dat získaných pomocí aktivity Data. Jednotlivé vzorky signálů mohou být identifikovány různými řády systému, uloženy a znovu identifikovány dle požadavků uživatele. Pro detailní popis správného postupu identifikace modelů systémů viz. 3.2 Identifikace.

Pro zvolení požadovaných dat k identifikaci slouží fragment Identifikace Hlavní (obr. 2.11). Můžeme zvolit ze záznamů signálů uložených v aktivitě Data nebo můžeme zvolit už dříve identifikovaný a uložený záznam signálu.



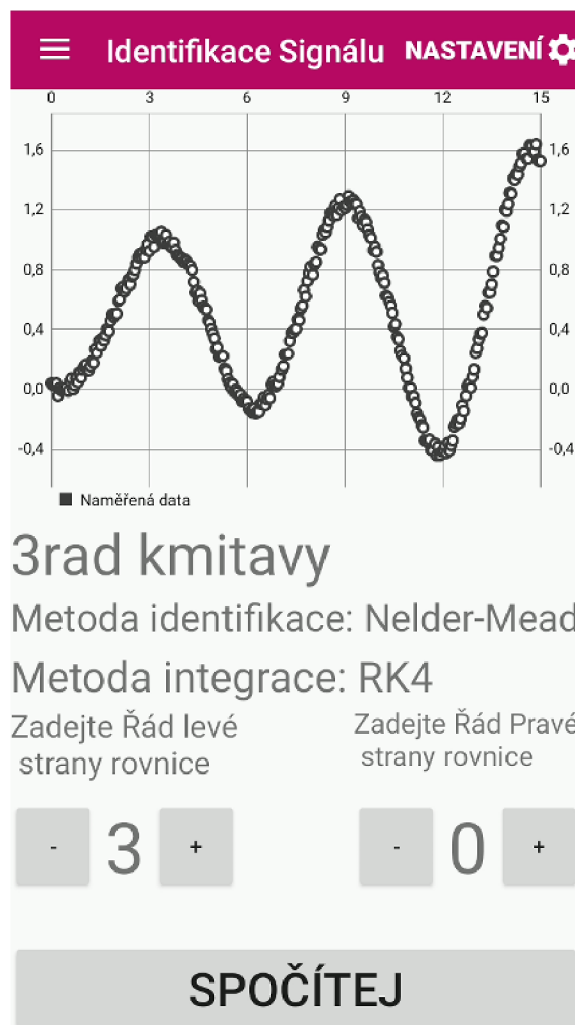
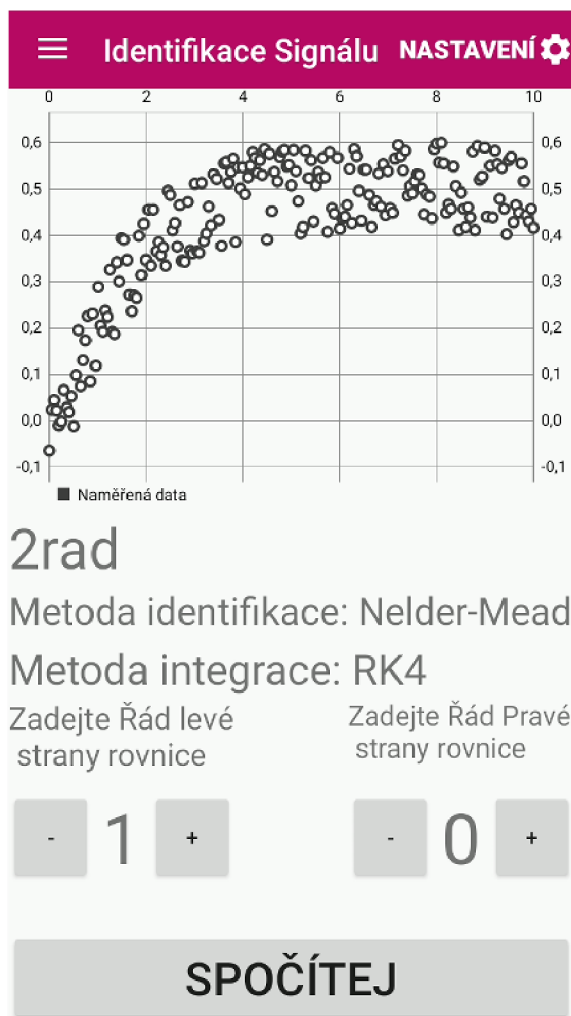
Obr. 2.10 Volba aktivity Identifikace



Obr. 2.11 Identifikace fragment hlavní

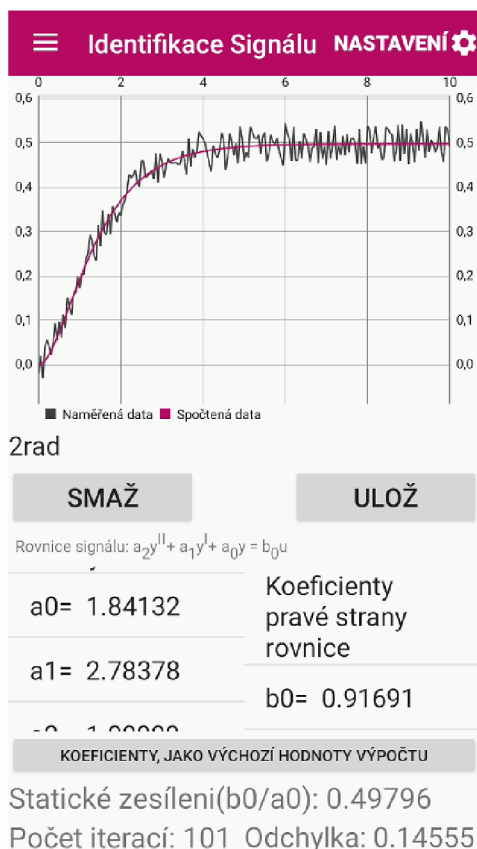
Po selekci vhodných dat se otevře fragment Identifikace, kde je nutno zvolit řád dynamického systému, resp. řád diferenciální rovnice s jakým chceme daná data aproximovat, a to pro levou i pravou stranu (obr. 2.12). Po stisknutí tlačítka spočítej je proveden výpočet a výsledek výpočtu je zobrazen na fragmentu Prohlížeč. Aplikace z praktických a užitných důvodů nepodporuje možnost identifikace systému vyššího řádu než osm. Zároveň je výběr řádu pravé i levé strany rovnice omezen tak, aby uživatel nebyl schopen zadat model systému a při tom porušit podmínku realizovatelnosti systému.

Fragment Prohlížeč prezentuje výsledek výpočtu identifikace dynamického systému zvoleným řádem formou grafu, kde jsou zpracovaná data proložena aproximační křivkou diferenciální rovnice. Samotná diferenciální rovnice se svými koeficienty je vypsána pod grafem (obr. 2.13). Pokud je uživatel s identifikací spokojen, může ji uložit do databáze uložených identifikací tlačítkem uložit. Pokud si přeje smazat některou z dříve uložených identifikací, tak jí načte z fragmentu hlavní a smaže tlačítkem smazat.

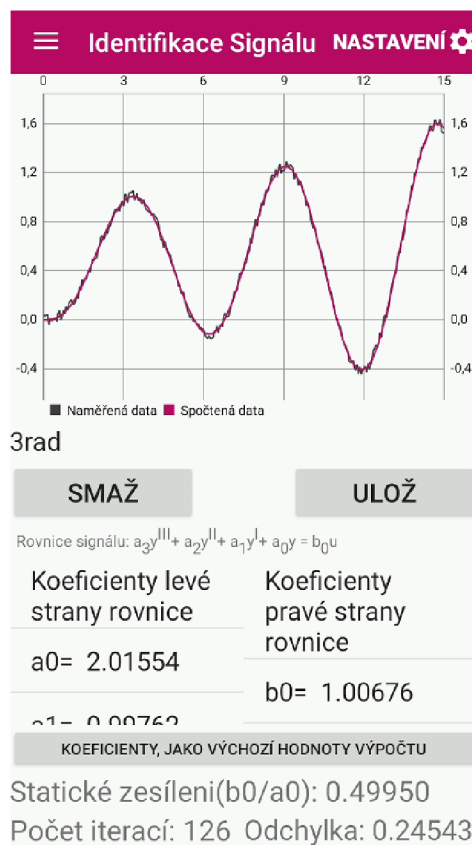


Obr. 2.12 Identifikace fragment identifikace

Mezi všemi fragmenty Aktivity Identifikace lze volně přecházet pomocí „swipe“, podobně jako u Aktivity Data. Pořadí fragmentů je: fragment Hlavní, fragment Identifikace, fragment Prohlížení. Používané výpočetní metody lze uživatelsky nastavit v menu nastavení – Metoda Identifikace a menu nastavení – Metoda Integrace (obr. 2.14 a obr. 2.15). Více detailních informací o nastavení metod minimalizace a integrace je k dispozici v kapitole 3.



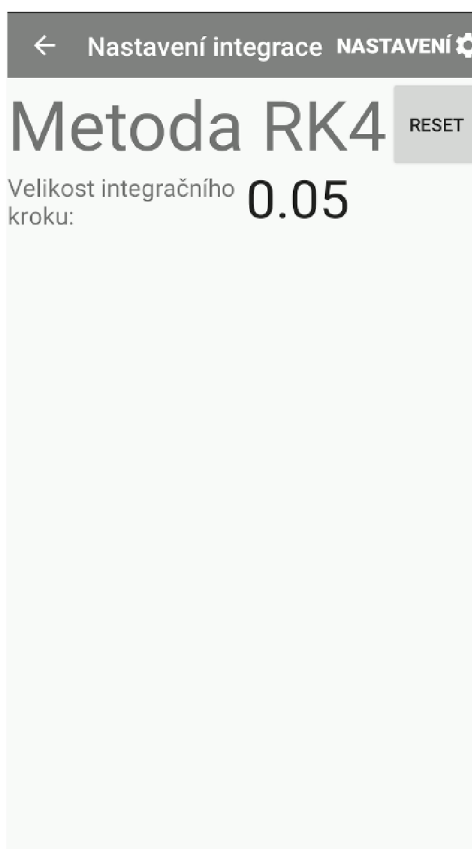
Obr. 2.13 Identifikace fragment Prohlížeč



Obr. 2.14 Identifikace nastavení minimalizace



Obr. 2.15 Identifikace nastavení integrace

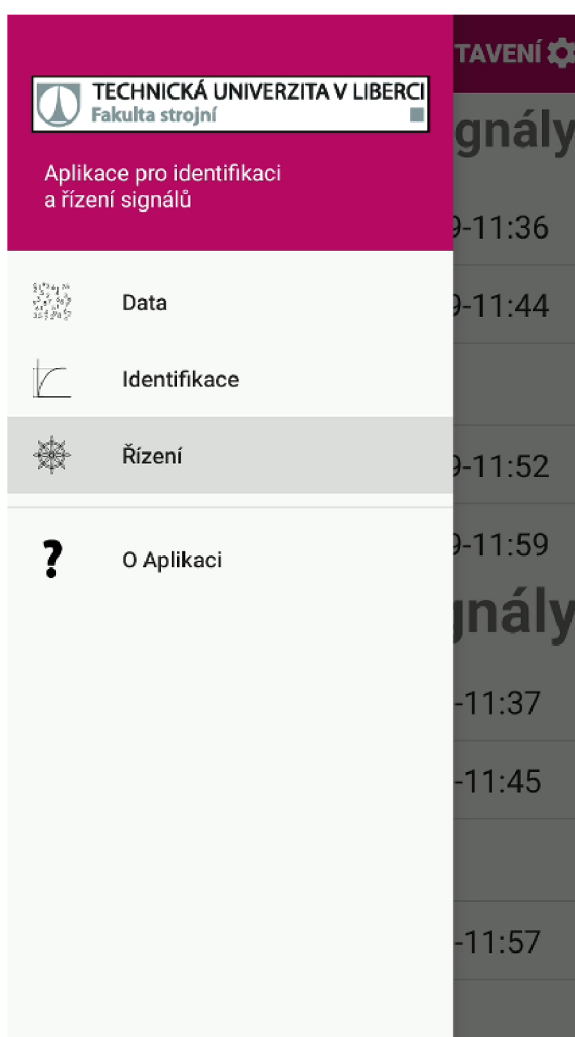


2.5 AKTIVITA ŘÍZENÍ

Úkolem aktivity Řízení je dle identifikovaného modelu systému, vytvořeného v aktivitě Identifikace, seřadit PID regulátor.

Volba žádaného modelu regulace probíhá ve fragmentu Řízení hlavní a je obdobná volbě měřeného signálu pro identifikaci systému. Můžeme zvolit identifikovaný model systému, uložený v aktivitě Identifikace, nebo již dříve uložený záznam řízení (obr. 2.17).

Po selekci vhodných dat se model systému načte do fragmentu řízení, kde je zobrazena jeho diferenciální rovnice (obr. 2.18). Dále je zde možno nastavit parametry seřízení, jako jsou požadovaná hodnota, a parametry časového horizontu řízení (začátek časového horizontu, konec časového horizontu, krok)⁵.



Obr. 2.16 Volba aktivity Řízení



Obr. 2.17 Řízení fragment Hlavní

⁵ Pro více informací o parametrech řízení viz. 3.3.3 Detailní postup při seřizování PID regulátoru

Obr. 2.18 Řízení fragment Řízení

Stisknutím tlačítka **SEŘIĎ REGULÁTOR** a výběrem požadovaného typu regulátoru (obr. 2.19) se provede automatické seřízení regulátoru. Na výběr je z celkem pěti typů regulátorů – P, PI, PID, PD a I. Více informací o regulování systémů je dostupných v sekci 3.3 Řízení.

Výsledek seřízení zvoleného typu regulátoru se zobrazí ve fragmentu prohlížení (obr. 2.20). Na seřízení regulátoru je možno dále provádět ruční úpravy změnou velikosti hodnot koeficientů K_c , T_i a T_d , jakožto měnit nastavení časového horizontu a žádané hodnoty regulace. Zadané změny jsou aplikovány tlačítkem **PŘEPOČÍTEJ**.

Všechny fragmenty aktivity jsou, jako u předchozí aktivity propojeny v pořadí: fragment Hlavní, fragment Řízení, fragment Prohlížeč, a lze se mezi nimi pohybovat pomocí přetažení „swipe“. Ukládání a mazání záznamů funguje stejně jako v předchozí aktivitě.

Aktivita Řízení, podobně jako aktivita Identifikace, nabízí možnost pokročilého nastavení metody minimalizace (ovlivňuje pouze automatické seřízení regulátoru (obr. 2.21)).

☰ Řízení Systému **NASTAVENÍ** ⚙️

PARAMETRY MODELU
3rad
Rovnice signálu: $a_3y^{(3)} + a_2y^{(2)} + a_1y^{(1)} + a_0y = b_0u$

Koeficienty levé strany rovnice	Koeficienty pravé strany rovnice
a0= 2.01554	b0= 1.00676
a1= 0.99762	
a2= 1.51407	

Regulátor typu P

Regulátor typu PI (a0): 0.49950

Regulátor typu PID 05

Regulátor typu PD 0

Regulátor typu I 0.0

Regulátor typu I 0

SEŘIĎ REGULÁTOR

☰ Řízení Systému **NASTAVENÍ** ⚙️

PARAMETRY MODELU
2rad
Rovnice signálu: $a_2y^{(2)} + a_1y^{(1)} + a_0y = b_0u$

Koeficienty levé strany rovnice	Koeficienty pravé strany rovnice
a0= 1.84132	b0= 0.91691
a1= 2.78378	
a2= 1.00000	

Regulátor typu P

Regulátor typu PI (a0): 0.49796

Regulátor typu PID 05

Regulátor typu PD 0

Regulátor typu I 0.0

Regulátor typu I 0

SEŘIĎ REGULÁTOR

Obr. 2.19 Řízení fragment řízení, výběr regulátoru

☰ Řízení Systému **NASTAVENÍ** ⚙️

3rad

SMAŽ ULOŽ

PARAMETRY REGULÁTORU 0.1

-	Kc:	2.24914	+
-	Ti:	1.58924	+
-	Td:	4.86242	+

Krok: 0.05
Počátek intervalu: 0.0
Konec Intervalu: 10.0
Žádaná hodnota: 1.0

PŘEPOČÍTEJ

☰ Řízení Systému **NASTAVENÍ** ⚙️

2rad

SMAŽ ULOŽ


PARAMETRY REGULÁTORU 0.1

-	Kc:	15.5852	+
-	Ti:	8.52697	+
-	Td:	9.11826	+

Krok: 0.05
Počátek intervalu: 0.0
Konec Intervalu: 10.0
Žádaná hodnota: 1.0

PŘEPOČÍTEJ

Obr. 2.20 Řízení fragment prohlížení

← Nastavení minimal... NASTAVENÍ 

Metoda Nelder-Mead RESET

Maximální počet iterací: **1000**

Požadovaná přesnost výpočtu: **0.01**

Výchozí hodnoty výpočtu (seed):

Vychozí hodnota P:0.1
Vychozí hodnota I:0.1
Vychozí hodnota D:0.1

Adaptivní parametry:

Koeficient Alpha: 1.0
Koeficient Beta: 1.0
Koeficient Gamma:0.75
Koeficient Delta: 1.0

Obr. 2.21 Řízení nastavení metod výpočtu

3 VÝPOČTOVÝ PRINCIP

Výpočtový aparát aplikace byl zvolen tak, aby poskytoval spolehlivé a přesné výsledky v co nejobecnějším formátu. Primární roli zde hrají optimalizační numerické metody, a numerické metody integrace. Přičemž jejich použití aktivitami Identifikace a Řízení se liší ve vstupních parametrech a mechanismu postupu výpočtu (obr. 3.2, obr. 3.13).

3.1 DYNAMICKÝ SYSTÉM

Existuje mnoho definic dynamických systémů, od volných slovních popisů, až po striktní matematické formulace. Pro potřebu této práce postačí definice dynamického systému jako objektu, ve kterém se mění hodnoty různých proměnných v důsledku časového vývoje a z něho plynoucích interakcí.

Příkladů takovýchto dynamických systémů lze nalézt bezpočet. Obecně by se dalo říci, že prakticky vše ve fyzickém světě, podléhá změnám v čase dle smysluplného konzistentního mechanismu, a proto lze vše z jednoho, nebo druhého úhlu pohledu považovat za dynamický systém.

Jako konkrétní příklady lze uvést: chování živých organismů (množení bakterií, etologie zvířat, růstové vzory rostlin, šíření určitého genu populací...), fyzikální principy v elektro a výpočetní technice (způsob šíření elektrického náboje, rychlost zpracování dat, přenos informace...), mechanické systémy (kmitání pružin, reakce na zatížení, převodové mechanismy...), astrofyzika a kosmologie (pohyby vesmírných těles, vyzařování hvězd...), a mnoho dalších.

V inženýrské kybernetice, nebo přesněji v teorii automatického řízení, však hlavním předmětem zájmu není, dynamický systém, jako takový, ale jeho interakce s vnějším světem. Tedy vstupy a výstupy ze systému a možnosti jejich regulace, popřípadě kontroly, pro dosažení požadovaných cílů.

Existují tři základní modely (myšlenkové koncepty) dynamických systémů, podle toho, jaké informace o pochodech uvnitř systému jsou známy:

a) Model bílé skřínky (White Box Model)

Všechny pochody, jež probíhají uvnitř systému jsou známy, systém je naprosto deterministický a neexistuje možnost výskytu náhodných jevů.

b) Model šedé skřínky (Grey Box Model)

Některé pochody, jež probíhají uvnitř systému jsou známy, systém je částečně deterministický, ale vyskytují se zde i náhodné jevy, nevysvětleny popisem systému. Prakticky všechny dynamické systémy reálného světa spadají do této kategorie. I ty nejdětalněji popsané systémy obsahují jisté procento stochastických (náhodných) jevů.

c) Model černé skříňky (Black Box Model)

O vnitřních pochodech, probíhajících uvnitř systému, není známo naprosto nic. Jediné, co je o tomto dynamickém systému známo, jsou jeho interakce s vnějším světem (obr 3.1).

V teorii automatického řízení se na dynamické systémy hledí především jako na modely černé skříňky, a tato práce se proto zabývá výhradně modelováním systému jako modelu černé skříňky. To znamená, že nás nezajímají detailní vnitřní pochody systému jako takového. Zajímají nás pouze vstupy a výstupy ze systému. Přesněji to, jakým způsobem výstup ze systému y reaguje na změnu vstupu do systému u . Dále se budeme zabývat již výhradně modely „černé skříňky“.



Obr. 3.1 Model černé skříňky

Jak je tedy patrné z obr. 3.1 výstup ze systému černé skříňky není nic jiného, než funkcí vstupu do systému

$$y = f(u). \quad (3.1)$$

Rovnice 3.1 platí pro statický či ustálený systém (převodní funkce vstupu a výstupu je konstantní), a má své využití například při zjišťování statické charakteristiky systému. Nicméně teorie automatického řízení se zabývá popisem dynamických systémů, tzn. převodní funkce se mění v čase t . Výstup je tedy funkcí vstupu a aktuální hodnoty času, rovněž vstup sám může být funkcí času

$$y(t) = f(u(t)). \quad (3.2)$$

K popisu závislosti vstupu a výstupu dynamického systému se dá přistupovat mnoha způsoby, kdy každý má své klady a zápory. Detailní rozbor těchto metodik není předmětem této práce pro více o informacích viz. seznam použité literatury [1],[3], [7], [8].

Pro matematický popis dynamických spojitých systémů v čase proměnných jevů se nejčastěji využívá obyčejná lineární diferenciální rovnice v obecném tvaru (rovnice 3.3). Jedná se z matematického hlediska o popis nejjednodušší a nejpřirozenější, kdy levá strana rovnice reprezentuje výstup ze systému, a pravá strana rovnice reprezentuje vstup do systému, řád diferenciální rovnice reprezentuje komplexnost interakcí jež v systému probíhají:

$$\begin{aligned} a_n y^n(t) + a_{n-1} y^{n-1}(t) + \dots + a_1 y'(t) + a_0 y(t) = \\ = b_m u^m(t) + b_{m-1} u^{m-1}(t) + \dots + b_1 u'(t) + b_0 u(t), \end{aligned} \quad (3.3)$$

kde: n – řád levé strany diferenciální rovnice, m – řád pravé strany diferenciální rovnice, a_0, \dots, a_n - koeficienty levé strany diferenciální rovnice, b_0, \dots, b_n - koeficienty pravé strany diferenciální rovnice.

Rovnice 3.3 je rovněž základním teorémem teorie automatického řízení spojitého dynamického systému a základní rovnicí, výpočtového mechanismu identifikace dynamického systému implementovaného v aplikaci.

Mnohem častěji se s ní však lze setkat ve tvaru Laplaceovy transformace (obraz funkce $f(t)$ při Laplaceově transformaci je funkce jedné komplexní proměnné s), jako s podílem Laplaceovy transformace výstupu $Y(s)$ ku vstupu $U(s)$ (rovnice 3.4). Tento tvar je nazýván obrazový přenos (značen $G(s)$, někdy $F(s)$). Jedná se o tvar vyskytující se ve většině odborné literatury a základní matematické východisko teorie automatického řízení:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0} \quad (3.4)$$

3.2 IDENTIFIKACE

Identifikace systému je tvorba matematického popisu systému na základě získaných hodnot.

3.2.1 Identifikace systémů obecně

Při spojitě identifikaci dynamických systémů, jde vždy o to, najít koeficienty vztahů, jež nejlépe popisuje vztah výstupu ze systému ku vstupu ze systému, a tedy nejlépe aproximuje naměřená data. Jinými slovy hledáme koeficienty obrazového přenosu.

Nejobecnějším tvar obrazového přenosu je popsán rovnicí 3.4. Tento obrazový přenos schopen popsat libovolný dynamický spojitý systém. V praxi se však často využívají speciální tvary obrazového přenosu, označované jako PROCESNÍ MODELY. Detailní rozbor těchto procesních modelů, jejich vlastnosti a omezení nejsou předmětem této práce, pro více informací viz. seznam použité literatury [1],[3], [7], [8]. Obecně platí, že jejich zavedení a používání v praxi má, či mělo různá praktická opodstatnění:

- redukce složitých modelů
- omezené množství parametrů – vyšší rychlost výpočtů
- potlačení, či vnucení kmitání modelu
- možnosti využívání tabelovaných hodnot, některých koeficientů
- využití grafických metod
- přímá implementace koeficientů do některých metod řízení
- a některé další...

Nejčastěji používané speciální struktury obrazových přenosů:

- a) Soustava prvního řádu s dopravním zpožděním T_d

$$G_1(s) = \frac{K}{T \cdot s + 1} \cdot e^{-s \cdot T_d}, \quad (3.5)$$

kde: T_d -dopravní zpoždění, K -statické zesílení (lze spočítat z rovnice 3.3 jako: b_0/a_0 , nebo odečíst jako ustálenou hodnotu (pokud je systém stabilní)), T -časová konstanta ($T \cong 0,63K$).

b) Soustava druhého řádu přetlumená

$$G_2(s) = \frac{K}{(T_1 \cdot s + 1)(T_2 \cdot s + 1)}, \quad (3.6)$$

kde: T_1, T_2 - jsou časové konstanty, které lze získat například z tabelovaných hodnot po odečtení z grafu.

c) N -tý řád kritické tlumení:

$$G_3(s) = \frac{K}{(T \cdot s + 1)^n}, \quad (3.7)$$

kde: n -je řád systému (řád levé strany diferenciální rovnice).

d) Třetí řád, kmitavá (tlumená) odezva:

$$G_4(s) = \frac{K}{(T_1 \cdot s + 1)[(T_0 \cdot s^2) + 2 \cdot \xi \cdot T_0 \cdot s + 1]}, \quad (3.8)$$

Kde: T_1, T_0 – jsou časové konstanty, ξ – součinitel tlumení systému.

e) Třetí řád, přetlumená odezva

$$G_5(s) = \frac{K}{(T_1 \cdot s + 1)(T_2 \cdot s + 1)(T_3 \cdot s + 1)}, \quad (3.9)$$

kde: T_1, T_2, T_3 - jsou časové konstanty.

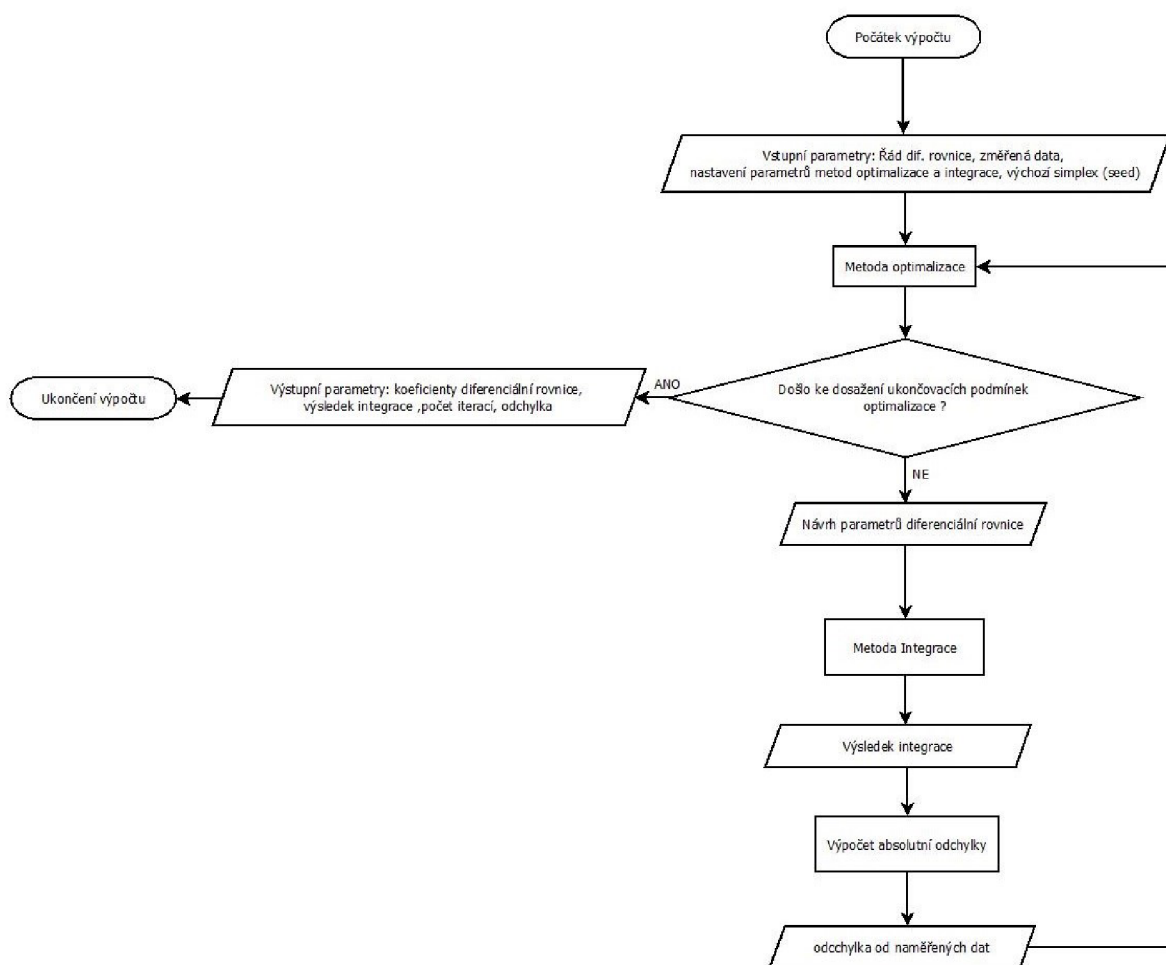
Všechny tyto rovnice vycházejí, přesněji řečeno jsou derivátem, rovnice 3.3, resp. rovnice 3.4. Pokud tedy máme k dispozici vyčíslenou rovnici obrazového přenosu v základním tvaru příslušného řádu, lze jí vždy převést požadovaný na procesní model. Pravdou však zůstává, že i když je možné v některých případech koeficienty procesních modelů dopočítat algebraicky (příklad a), či odečíst z grafu naměřených hodnot (příklad b). V drtivé většině případů se to neprovádí a pracuje se s nimi pomocí numerických metod optimalizace, kdy jsou jednotlivé koeficienty redukovány na prosté parametry příslušné optimalizační funkce.

3.2.2 Implementace v Aktivitě identifikace

Výpočtový mechanismus identifikace je v Aplikaci realizován rovněž na principu výpočtu koeficientů obrazového přenosu optimalizační numerickou metodou minimalizace odchylky funkce. Vzhledem k tomu, že od obrazového přenosu nejsou vyžadovány žádné speciální vlastnosti, kromě toho, aby byl co nejobecnější a popisoval co nejširší množinu dynamických systémů co nejflexibilněji a zároveň nejpresněji. Je vhodné pracovat s obrazovým přenosem v základním obecném tvaru (rovnice 3.4), přesněji řečeno lze numericky dopočítávat přímo koeficienty diferenciální rovnice 3.3. Implementace obrazových přenosů různých procesních modelů je dobrým námětem na rozšíření této práce. Schématický postup výpočtu aktivity Identifikace je znázorněn na obr. 3.2.

Výpočet identifikace systému je započat vložení výchozích dat výpočtu: Řád diferenciální rovnice (levé i pravé strany), změřená vstupní data, nastavení parametrů metod výpočtu, výchozí hodnoty simplexu. Následuje zpracování těchto dat metodou optimalizace, která navrhne počáteční řešení diferenciální rovnice⁶. Toto řešení je předáno metodě integrace, která provede integraci této diferenciální rovnice⁷. Z výsledku integrace je následně počítána absolutní odchylka dle rovnice:

$$absolutní\ odchylka = e_{abs} = \sum_{i=0}^n |y_i\ měřena - y_i\ spočtená| \quad (3.10)$$



Obr. 3.2 Diagram mechanismu výpočtu identifikace v Aktivitě Identifikace

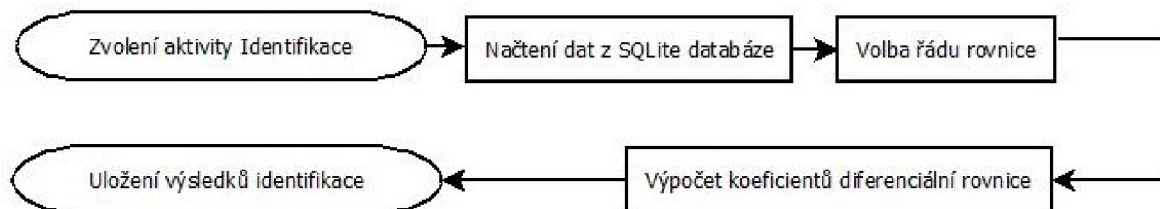
Takto spočtená odchylka se vrací do metody optimalizace, kde je posouzeno, jestli změna simplexu zlepšila řešení. Algoritmus je ukončen, pokud se hodnota odchylky již nezlepšuje dle zadané přesnosti, nebo pokud došlo k maximálnímu počtu iterací. Pokud je některá z těchto podmínek splněna, dochází k ukončení algoritmu výpočtu. Návrhovými hodnotami z výpočtu jsou: koeficienty diferenciální rovnice, spočtené hodnoty integrace v jednotlivých bodech, počet provedených iterací a dosažená hodnota absolutní odchylky.

⁶ Pro více informací o implementované metodě optimalizace viz.3.4.

⁷ Pro více informací o implementované metodě integrace viz. 3.5.

3.2.3 Detailní postup při identifikaci

Praktický postup identifikace byl již zevrubně popsán v kapitole 2.4. Standardně pobíhá postup identifikace dle schématu (obr. 3.3). Provedeme načtení požadovaných dat ve fragmentu hlavní → zvolíme požadovaný řád diferenciální rovnice → provedeme výpočet.



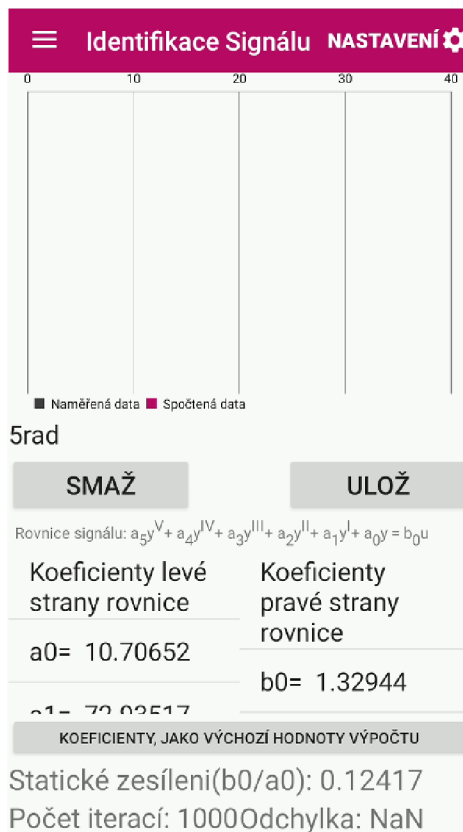
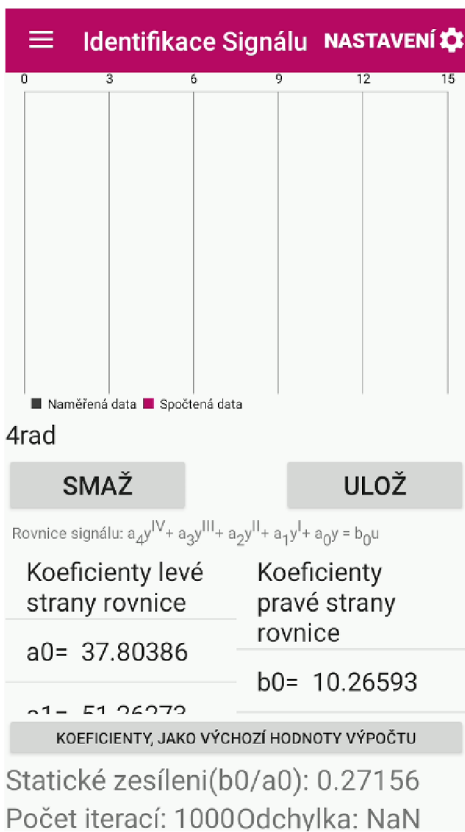
Obr. 3.3 Schéma základní identifikace modelu spojitého dynamického systému

Přičemž platí, že vyšší řád diferenciální rovnice znamená detailnější popis systému. Pro většinu aplikací však stačí identifikace systému prvním, popřípadě druhým řádem. Konvergence optimalizační metody je v těchto případech prakticky jistá. Nicméně se lze dostat do situace, kdy matematický model prvního či druhého řádu není dostačující.

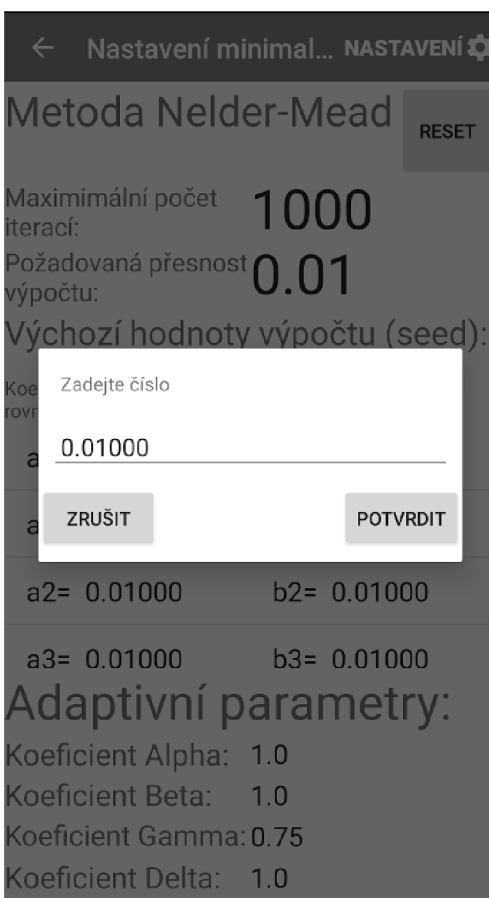
V těchto případech, kdy je použit řád modelu vyšší než jedna či dva. Není konvergence implementované optimalizační metody (Nelder-Mead) zaručená, a velice často se stává, že výpočet „spadne“ do lokálního minima funkce obr. 3.4. Pravděpodobnost takovéto situace strmě stoupá s velikostí vstupního vektoru optimalizační funkce (řádem diferenciální rovnice).

Tento problém lze do jisté míry obejít správným (nebo alespoň jiným) nastavením vektoru vstupních parametrů do optimalizační metody. To lze provádět ručně v nastavení minimalizace (obr. 3.5), pokud jsou známy přibližné hodnoty, které by dané koeficienty měly nabývat. Pro praktické využití, je však tento postup zdlouhavý a pracný, neboť volba těchto parametrů se podobá ve většině případů házení hrací kostkou a „doufání“ v optimální výsledek.

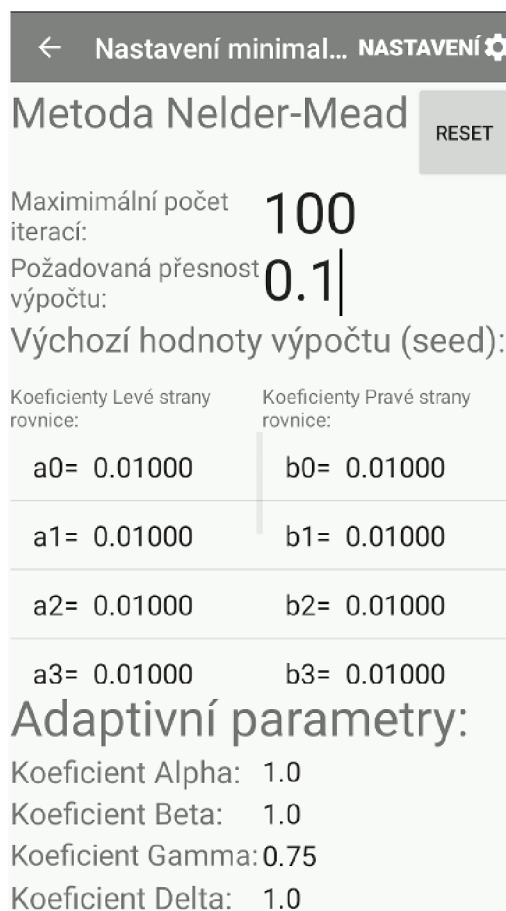
Praktické řešení je provést daný výpočet s výchozími parametry a nechat metodu zkonvergovat do lokálního minima. Pro rychlejší pokusné výpočty je vhodné snížit počet iterací a požadovanou přesnost výpočtu (obr. 3.6). Výpočet pak probíhá rychleji i pro systémy vyššího řádu a průběžně si ukládat mezivýsledky. Poté vzít tyto neplatné koeficienty a použít je jako výchozí vektor další optimalizace pomocí tlačítka KOEFICIENTY, JAKO VÝCHOZÍ HODNOTY VÝPOČTU (obr. 3.7). S tímto mechanismem lze po několika minutách experimentování dosáhnout kýžených výsledků identifikace systému i pro systémy vyšších řádů (obr. 3.8). Výpočet zde probíhá dle schématu zobrazeném na obr. 3.9.



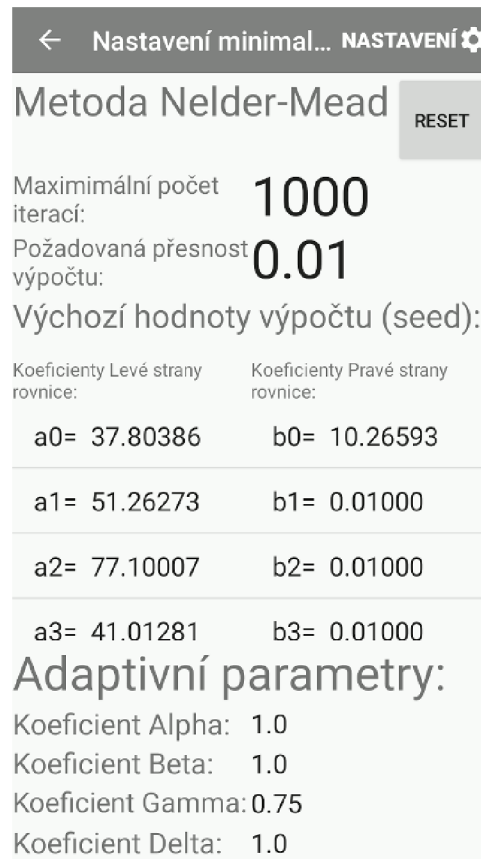
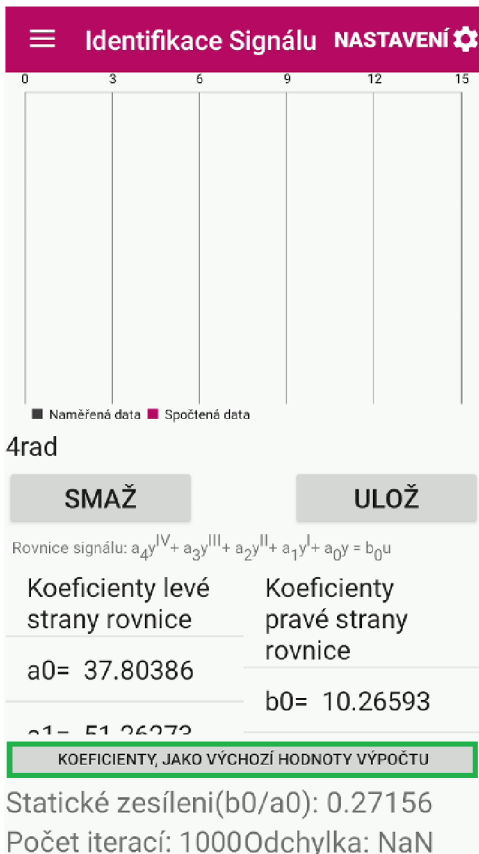
Obr. 3.4 Příklady špatné konvergence metody optimalizace



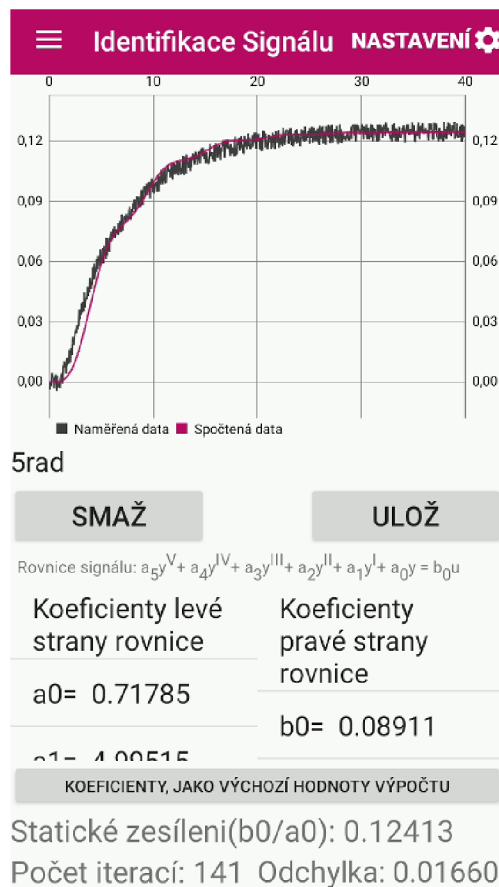
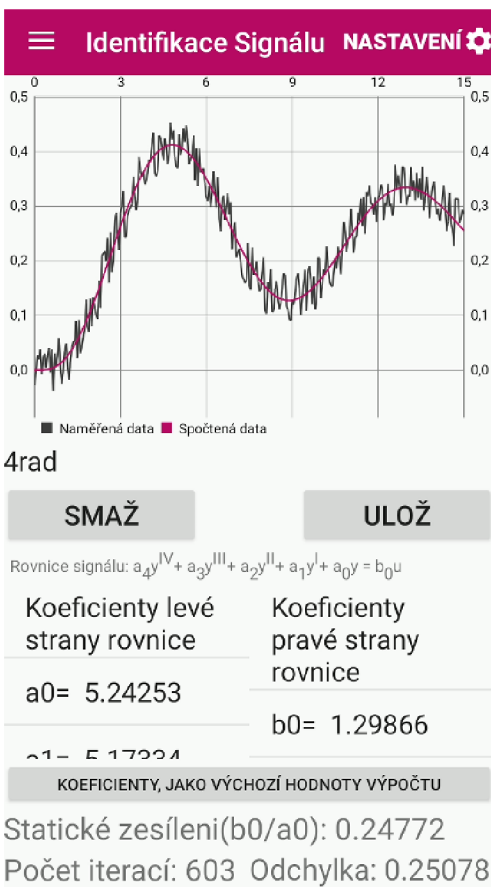
Obr. 3.5 Nastavení výchozích parametrů metody minimalizace



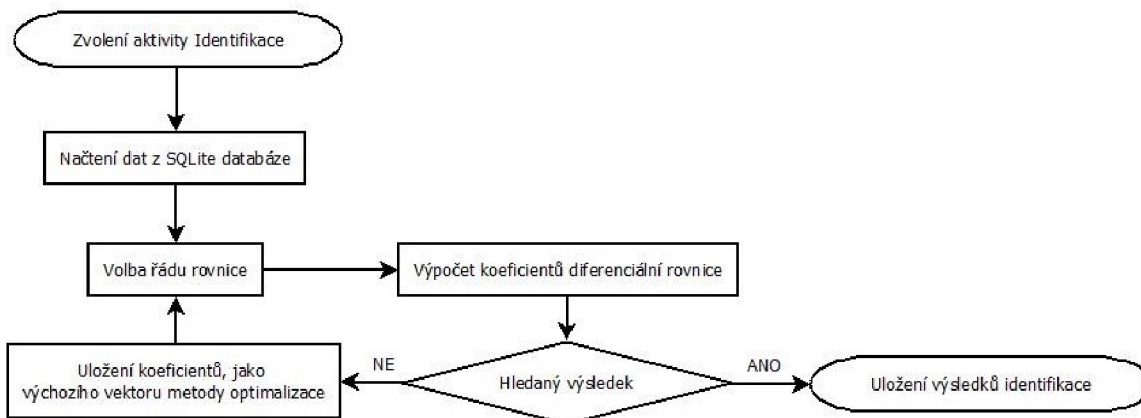
Obr. 3.6 Snížení kritérií pro ukončení výpočtu



Obr. 3.7 Nastavení koeficientů, jako výchozích parametrů pro další výpočet



Obr. 3.8 Identifikace systémů vyšších řádů



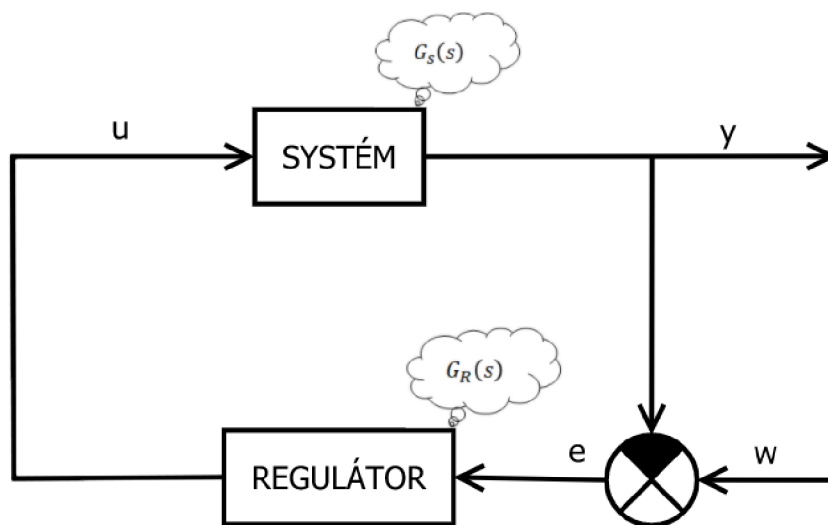
Obr. 3.9 Algoritmus identifikace systémů vyšších řádů

Je nutné si uvědomit, že toto je čistě inženýrský přístup řešení problému, nikoli matematický. Pro jeho sofistikovanější matematické řešení by bylo nutno značně modifikovat celý výpočtový aparát, přičemž je mít nutno na paměti, že neexistuje ideální optimalizační metoda, která konverguje vždy a za všech podmínek.

3.3 ŘÍZENÍ

Základní myšlenkou použití PID regulátoru, resp. řízení, nějaké veličiny obecně, je dosažení požadované hodnoty, v požadovaném čase, požadovaným způsobem (ku příkladu je vyžadováno, aby změna probíhala co nejrychleji, nebo bez překmitu výstupní veličiny apod.). Obecný princip regulace, pomocí regulátoru vychází ze základního zapojení zpětnovazebné smyčky (obr. 3.10). Kdy regulátor nastavuje veličinu u vstupující do systému, na základě hodnoty regulační odchylky e . Hodnota regulační odchylky e , je přímo závislá na hodnotě výstupu ze systému y a hodnotě poruchové veličině w (žádaná hodnota) vztahem 3.11. Takový systém je poté popsán obrazovým přenosem $G_{wy}(s)$, mezi výstupní y a žádanou hodnotou w , vztah 3.12.

$$e = w - y. \quad (3.11)$$



Obr. 3.10 Základní regulační smyčka regulátoru

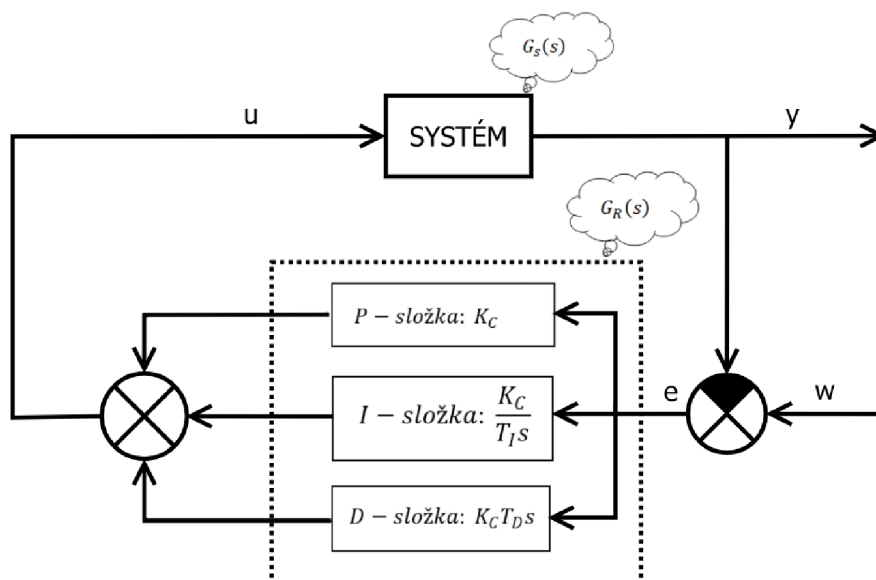
$$G_{wy} = \frac{G_R(s)G_S(s)}{1 + G_R(s)G_S(s)}, \quad (3.12)$$

kde: $G_S(s)$ -obrazový přenos systému, $G_R(s)$ -obrazový přenos regulátoru.

3.3.1 Princip PID regulátoru

Proporcionálně integračně derivační regulátor, dále pouze PID regulátor, představuje matematicky jednoduchý, výpočetně nenáročný, časem prověřený typ spojitého regulátoru, se kterým je obeznána drtivá většina technické veřejnosti, a který je do dnešních dnů využíván ve většině aplikací automatického řízení. Skládá se ze tří základních složek:

- a) Proporcionální složky P,
- b) Integrační složky I,
- c) Derivační složky D.



Obr. 3.11 Schématické zapojení PID regulátoru do regulační smyčky

Suma těchto složek dohromady poté tvoří samotný PID regulátor, schematicky znázorněn na obr. 3.11, kde je substituován do základního regulačního schématu. Obrazový přenos PID regulátoru, v Laplaceově transformaci je poté popsán rovnicí 3.13. To, že je PID regulátor sumou jednotlivých složek, má v praxi velký význam neboť ze základního PID regulátoru lze snadným vyřazením příslušných složek vytvořit všechny jeho varianty (P, PI, PD, I).

$$G_R(s) = K_C \left(1 + \frac{1}{T_I s} + T_D s \right), \quad (3.13)$$

kde: K_C -zesílení regulátoru (proporcionální konstanta), T_I -integrační konstanta, T_D -derivační konstanta. Zpětnou Laplaceovou transformací rovnice 3.13 obdržíme vztah mezi vstupem do PID regulátoru (regulační odchylkou e) a výstupem z něj veličinu u , jako funkci času:

$$u(t) = K_C \left(e(t) + \frac{1}{T_I} \int_0^t e(t') dt' + T_D \frac{de(t)}{dt} \right). \quad (3.14)$$

Pro praktickou implementaci za použití výpočetní techniky, je dále nutno rovnici 3.14 přepsat jako funkci času dle vztahu 3.15, pomocí disktrétních hodnot:

$$t = k \cdot t_s, \quad (3.15)$$

kde: t_s -velikost časového kroku výpočtu, k -krok regulátoru $k = 0,1,2,3 \dots$

tedy

$$u(KT_s) = K_C \left[e(KT_s) + \frac{T_s}{T_I} \sum_{i=0}^K e(iT_s) + T_D \frac{e(KT_s) - e[(K-1)T_s]}{T_s} \right], \quad (3.16)$$

resp.

$$u(KT_s) = K_C e(KT_s) + K_C \frac{T_s}{T_I} \sum_{i=0}^K e(iT_s) + K_C T_D \frac{e(KT_s) - e[(K-1)T_s]}{T_s}, \quad (3.17)$$

Tvar rovnice PID regulátoru dle rovnice 3.17 je velice důležitý, protože na jedné straně umožňuje pracovat s časem jako s disktrétní hodnotou (ve výpočetní technice se pracuje zpravidla pouze s disktrétními hodnotami), a na straně druhé separuje jednotlivé složky PID regulátoru. Jedná se tedy o tvar obrazového přenosu PID regulátoru, prakticky implementovaný v aplikaci.

Jak již bylo řečeno PID regulátor, je regulátorem spojitým, to znamená, že ho nelze prostou numerickou aproximací nikdy zcela přesně simulovat (řešení je zatíženo chybou jež vzniká v důsledku nespojitě zpětné vazby regulátoru). Nicméně při vhodné volbě velikosti výpočetního kroku t_s , se lze k přesnému řešení přiblížit na tolik, že je daná simulace použitelná pro všechny praktické účely⁸.

Pro praktické používání aplikace, je nutno si uvědomit, že výpočet probíhá po jednotlivých integračních krocích t_s , viz. výše. Přičemž počet těchto kroků, které musí zařízení spočítat, se řídí vztahem:

$$n \cong \frac{(t - t_0)}{t_s}, \quad (3.18)$$

kde: n - počet integračních kroků, t -konec časového horizontu, t_0 -počátek časového horizontu (v současnosti vždy nula, hodnota počátek v aplikaci slouží pouze pro zoom grafu na žádanou oblast), t_s -velikost časového kroku výpočtu.

Počet integračních kroků n přímo ovlivňuje výpočetní náročnost daného výpočtu, jeho číslo lze snížit, buďto zvětšením kroku t_s , nebo zkrácením časového horizontu, zmenšením hodnoty t . To je na jednu stranu obzvláště důležité pro využívání aplikace na starších, a pomalejších zařízeních, jelikož se může snadno stát, že nastavíme příliš velký počet n , v jehož důsledku může aplikace „zamrznout“. Na druhou stranu nám to umožňuje skutečně využít výpočtový potenciál rychlých a výpočetně silných zařízení (pokud chceme simulovat dlouhý časový horizont s velkou přesností apod.). Volba t_s , stejně jako t , je v současné době na uživateli aplikace, nicméně v budoucnu by bylo dobré omezit tyto hodnoty dle

⁸ Pro výsledky praktických výpočtů, s různými kroky viz. 4.3 Verifikace seřizování PID regulátoru

specifikací konkrétních zařízení. Pro výsledky praktických výpočtů na reálném zařízení viz. 4.3 Verifikace a seřizování PID regulátoru.

3.3.2 Princip seřizování regulačního obvodu aplikací

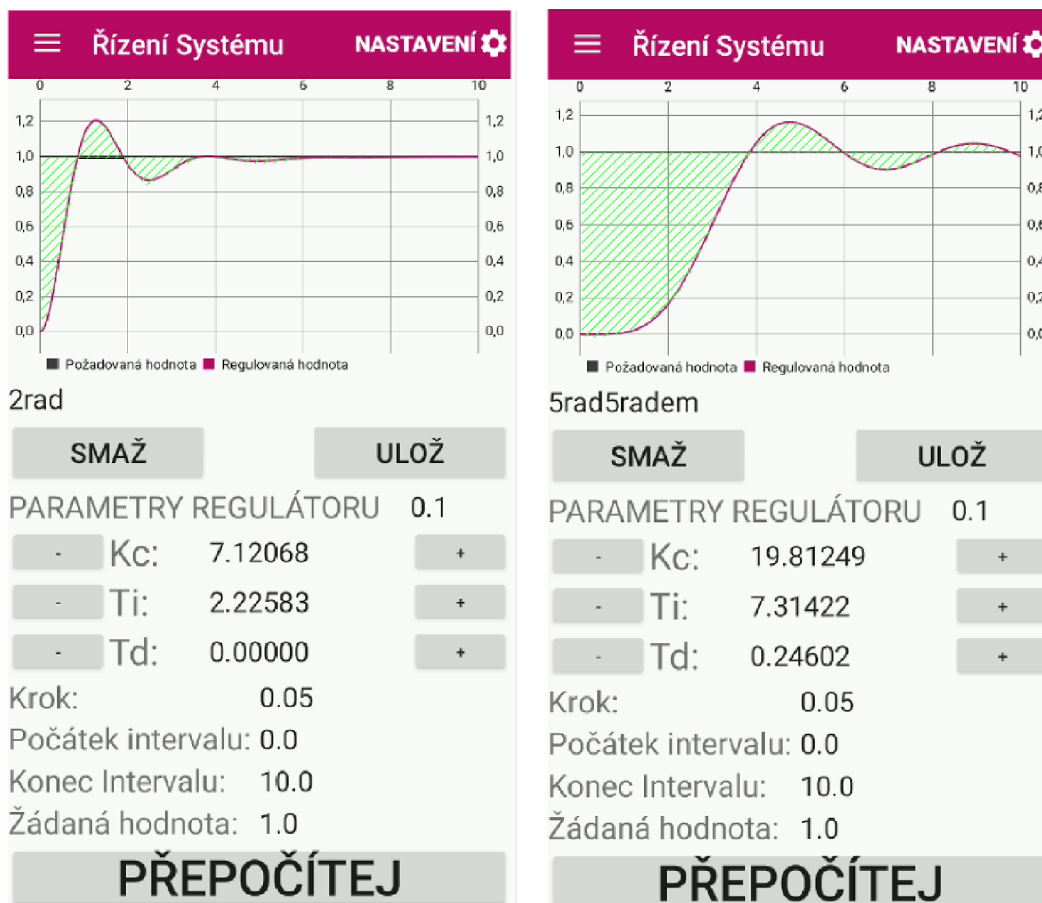
Při seřizování regulátoru PID, nebo některé z jeho modifikací, je nutno najít optimální hodnoty příslušných konstant K_C , T_I a T_D (pro více informací viz. 3.3.3). Pro nalezení optimálních hodnot těchto parametrů existuje řada přístupů, od takřka empirických, přes sofistikovanější metody s tabelizovanými hodnotami, až po sofistikované metody autotuningu. Přičemž jejich použití, praktičnost a spolehlivost, je často závislá na aplikaci a žádaném výsledku. Implementace takového množství algoritmů a nastavení je, bohužel, nad možnosti této práce, proto aplikace obsahuje pouze jedinou metodu seřízení regulátoru – metodu minimalizace absolutní integrační odchylky. Implementace dalších mechanismů automatického seřízení však dozajista představuje vhodné budoucí rozšíření této práce.

Vzhledem k tomu, že aplikace má být co nejuniverzálnější, je zde metoda automatického seřízení implementována jako „soft mechanika“, která má uživateli navrhnout „prvotní“ řešení. Je ale na uživateli samotném, jestli toto řešení přijme, nebo se rozhodne pro další úpravy regulačních veličin.

Samotná mechanika seřizování parametrů PID regulátoru není nepodobná mechanice hledání koeficientů při identifikaci modelu systému viz. 3.2.2, i zde se dochází k minimalizaci odchylky, v tomto případě se však nejedná o minimalizaci sumy odchylek spočtených dat od naměřených.

Nýbrž o sumu regulačních odchylek e_i regulované hodnoty y_i od hodnoty požadované w_i , při daném nastavení PID regulátoru. Matematicky je tento vztah popsán rovnicí 3.18. Graficky je daná suma odchylek reprezentována zvýrazněnou plochou na obr. 3.12.

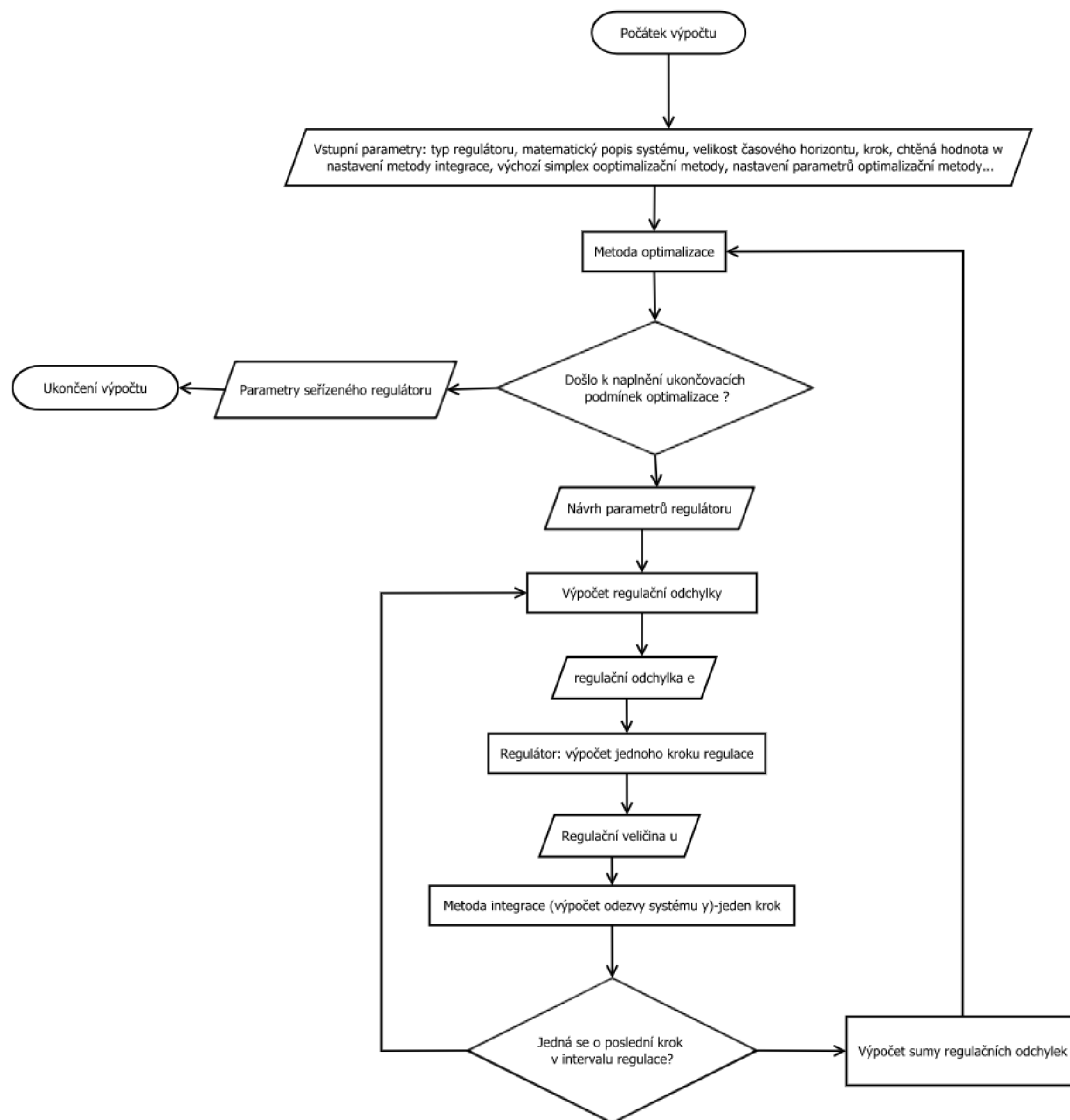
$$\text{regulační odchylka} = e_{regul} = \sum_{i=0}^n |w_i - y_i \text{ spočtená odezva}|. \quad (3.19)$$



Obr. 3.12 Řízení-plocha regulační odchylky e

Algoritmus celého výpočtu je schematicky zobrazen na obr. 3.13. Algoritmus začíná prvotním návrhem parametrů PID regulátoru (to, které parametry jsou navrhovány, záleží na uživatelské volbě regulátoru). Následuje spočtený regulačního výpočtu s tímto nastavením regulátoru. To probíhá v zásadě dle schématu obr. 3.11. Nejprve je spočtena regulační odchylka e (rovnice 3.11) rozdílem žádané hodnoty regulované veličiny w a hodnoty y . Tato hodnota vstupuje do regulátoru, kde je z něj spočten jeden krok výstupní veličiny u . Ten je následně vyslán na vstup do systému (jehož matematický popis je znám, viz. kapitola 3.2 identifikace), kde je z něj spočtena metodou numerické integrace výstupní veličina y pro další krok. Celý postup regulace je opakován, dokud nejsou zpracovány všechny kroky v zadaném časovém horizontu. Poté je spočtena regulační odchylka dle vztahu 3.18, která je zpracována optimalizační metodou, ta buďto navrhne nové parametry seřízení regulátoru, nebo celý výpočet ukončí (v případě dosažení podmínek ukončení výpočtu).

Praktická implementace je provedena v aplikaci řadou metod, které se navzájem volají a ve výsledku tvoří dva vnořené cykly (obr. 3.13).

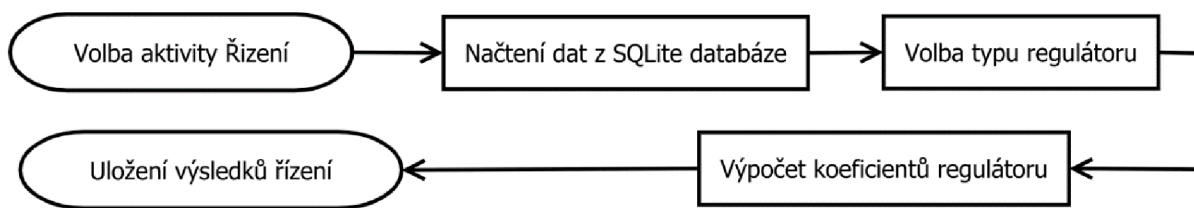


Obr. 3.13 Schéma algoritmu seřizování regulátoru

Jako v případě identifikace se může stát, že optimalizační metoda dokonverguje do lokálního minima. To se zde stává mnohem řídkěji než v případě aktivity identifikace. Jelikož optimalizační metoda má maximálně, v případě PID regulátoru, najít hodnoty tří parametrů. Pokud se to však stane rostou hodnoty grafu seřízení nade všechny meze (graf regulace se vůbec nevykreslí). Nebo nedojde k ustálení regulačního procesu. Řešení takového problému je buďto změnit výchozí parametry optimalizační metody (nastavení → metoda minimalizace → výchozí hodnota P, I, D), změnit typ regulátoru pro automatické seřízení nebo nastavit parametry regulátoru ručně.

3.3.3 Detailní postup při seřizování PID regulátoru

Praktický postup regulace byl již zevrubně popsán v kapitole 2.5. Standardně pobíhá postup regulace z uživatelského pohledu dle schématu na obr. 3.14.



Obr. 3.14 Schéma základního seřízení PID regulátoru

V technické praxi je však často nutno regulační pochody „doladit“ dle specifických požadavků. Tato funkcionality je jednou z hlavních předností této aplikace. Jak bylo zmíněno funkce automatického seřízení regulátoru je implementována pouze jako „soft“ mechanika, a uživatel má možnost přenastavit veškeré parametry regulátoru ručně (obr. 3.15). Změny jsou aplikovány po stisknutí tlačítka přepočítej a výsledek je ihned zobrazen na hlavním grafu. Aplikace obsahuje, kromě možnosti zadat přímo požadovaná čísla ručně, i funkcionality inkrementálního nastavování hodnot. To je výhodné obzvláště pokud chceme „ladit“ daný regulační proces (neboť ruční zadávání čísel je v tomto případě záležitostí více než únavnou). Hodnota inkrementu je rovněž nastavitelná uživatelem.

The screenshot displays the application's manual tuning interface. On the left, the 'PARAMETRY MODELU' section shows the signal equation $a_2y'' + a_1y' + a_0y = b_0u$ and lists coefficients: $a_0 = 4.79371$, $a_1 = 7.41784$, $a_2 = 2.72719$, $b_0 = 2.43189$. Below this, a dropdown menu for 'Regulátor typu' is open, showing options for P, PI, PID, PD, and I. The 'SEŘIŘ REGULÁTOR' button is visible at the bottom of this panel.

The middle and right panels show the 'PARAMETRY REGULÁTORU' for a 0.1 step. Each panel includes a graph of the control signal (red line) and the regulated signal (black line) over time. The middle panel shows the current parameters: $K_c = 7.12068$, $T_i = 2.22583$, $T_d = 0.00000$. The right panel shows the target parameters: $K_c = 1$, $T_i = 1$, $T_d = 0.00000$. The 'PŘEPOČÍTEJ' button is highlighted in green in both panels.

Obr. 3.15 Ruční seřizování PID regulátoru

Jak již bylo řečeno v 3.3.1, přesnost seřízení je do značné míry ovlivněna velikostí kroku integrace, při praktických výpočtech se jako optimální velikost kroku t_s jevila hodnota 0.001 viz 4.3. Praktickou pomůckou pro ověření správnosti seřízení je, po dokončení seřizování, si takto získaný průběh ověřit snížením kroku o jeden desetinný řád (výpočet může trvat i značnou chvíli – opět záleží na konkrétním zařízení, a délce časového horizontu t), nicméně pokud se průběh grafu nijak významně nezmění, lze výsledek považovat za validní.

Dobré seřízení PID regulátoru je do jisté míry otázka profesní zkušenosti a experimentování (hlavní účel této práce, resp. aplikace). Nicméně pro erudované seřizování je nutno znát základní funkce jednotlivých složek PID regulátoru, a to, jak ovlivňují regulační pochod. Detailní popis veškerých nuancí, které zahrnují PID regulaci, je nad rámec této práce. Krátký popis prezentovaný níže lze považovat pouze za hrubý nástin regulačních mechanik.

Při seřizování PID regulátoru, se seřizují pouze tyto tři konstanty K_C , T_I , T_D (rovnice 3.17), kdy každá z nich má specifický účinek na regulační pochod:

- **Proporcionální konstanta K_C :** Konstanta K_C funguje jako prostý zesilovač, přičemž akční veličina je v tomto případě přímo úměrná regulační odchylce, tedy urychlení náběhu regulačního děje. Avšak její velká hodnota může mít za následek kmity systému. Je nutné mít na paměti, že při použití samotné regulační konstanty K_C nemůže regulovaná veličina statického systému dosáhnout přesné požadované hodnoty, jelikož zde vždy bude takzvaná trvalá regulační odchylka. Ta vychází ze vztahu 3.12 do kterého se dosadí vztah 3.13 (pouze proporcionální složka). Po dosazení konkrétních hodnot do těchto vztahů, je zřejmé, proč má regulátor typu P trvalou regulační odchylku

$$G_{yw}(s) = \frac{K_C \frac{1}{s^2 + 3s + 2}}{1 + K_C \frac{1}{s^2 + 3s + 2}} = \frac{K_C}{s^2 + 3s + 2 + K_C} \rightarrow \quad (3.20)$$

$$\rightarrow \text{statické zesílení } \left(\frac{b_0}{a_0}\right) = \frac{K_C}{2 + K_C} < 1,$$

- **Integrační konstanta T_I :** Integrační konstanta představuje integrál sumy odchylek. Často do jisté míry zpomaluje regulační proces. Také může budit překmity systémů nad požadovanou hodnotu. **Složkou I se dá odstranit trvalá regulační odchylka statického systému**

$$G_{yw}(s) = \frac{\left(K_C + \frac{K_C}{T_I s}\right) \cdot \left(\frac{1}{s^2 + 3s + 2}\right)}{1 + \left(K_C + \frac{K_C}{T_I s}\right) \cdot \left(\frac{1}{s^2 + 3s + 2}\right)} = \frac{K_C T_I s + K_C}{T_I s^3 + 3T_I s^2 + (2 + K_C)T_I s + K_C} \rightarrow \quad (3.21)$$

$$\rightarrow \text{statické zesílení } \left(\frac{b_0}{a_0}\right) = \frac{K_C}{K_C} = 1$$

- **Derivační konstanta T_D :** Derivační konstanta především tlumí kmity, resp. tlumí odezvu celého systému. Praktické použití má především při seřizování kmitavých soustav, u kterých nedochází k samovolnému ustálení.

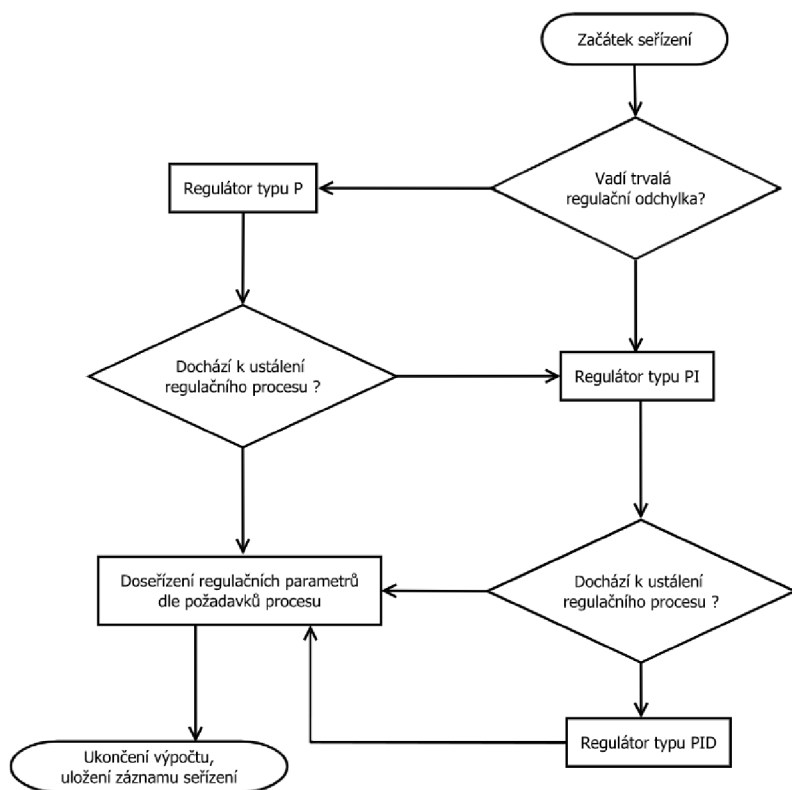
$$G_{yw}(s) = \frac{\left(K_C + \frac{K_C}{T_I s} + K_C T_D s\right) \cdot \left(\frac{1}{s^2 + 3s + 2}\right)}{1 + \left(K_C + \frac{K_C}{T_I s} + K_C T_D s\right) \cdot \left(\frac{1}{s^2 + 3s + 2}\right)} \rightarrow \quad (3.22)$$

$$\rightarrow \text{statické zesílení } \left(\frac{b_0}{a_0}\right) = \frac{K_C}{K_C} = 1$$

Orientační algoritmus postupu seřizování PID regulátoru, je zobrazen na obr. 3.16. Pokud je irelevantní trvalá regulační odchylka, je možné seřizovat pouze P složku PID regulátoru. V praxi však je nejčastěji používaným regulátorem regulátor PI (je doporučováno začínat rovnou seřizováním PI regulátoru). Pokud nelze dosáhnout ustálení systému pomocí PI regulátoru, je vhodné vyzkoušet plný PID regulátor (nestabilní soustavy vyššího řádu se silnými oscilacemi často vyžadují D složku).

V praxi jsou nejčastěji řešeny dva typy požadavků na seřízení PID regulátoru. Prvním je rychlost procesu řízení, kdy jde o to, aby bylo dosaženo požadované hodnoty v daném. Druhým je požadavek na minimalizaci či eliminaci překmitu řízené veličiny (například nechceme, aby souřadnice CNC obráběcího stroje překročila jistou hodnotu – výroba zmetku apod.).

Obecně k urychlení regulačního procesu, bez obměny hardwaru, využívá zvýšení P-složky regulátoru (nicméně za cenu překmitu a poněkud dramatického přechodu regulované veličiny). Překmitu se dá naopak zamezit snížením P-složky, resp. zvýšením I-složky či D-složky regulátoru. Jak již bylo zmíněno seřizování PID regulátorů je velice specifická tematika, která se dramaticky liší při regulaci každého systému. Zde opět platí, již několikrát zmíněné, že experimentování je klíčové.



Obr. 3.16 Diagram seřizování PID regulátoru

Regulátory typu PD a I. Jsou speciální regulátory, využívané, v přetlumených soustavách (PD), nebo v soustavách se silně buzenými kmity (I), jejich využití je specifické. A záleží čistě na uživateli, zda se je rozhodne využívat, či co je spíše pravděpodobné, nebude mít pro daný systém jiné východisko.

3.4 OPTIMALIZAČNÍ METODA FLEXIBILNÍHO SIMPLEXU

Metoda flexibilního simplexu obecně známější jako metoda „Nelder-Mead“ dále pouze metoda NM, popř. „metoda svažujícího se simplexu“. Je klasická metoda lokální optimalizace, která nevyužívá derivací při hledání minima. Představena světu byla svými autory: Johnem Nelderem a Rogerem Meadem již v roce 1965 v jejich práci „A simplex method for function optimization“. V inženýrské praxi je jednou z nejvyužívanějších, a nejoblíbenějších metod hledání minima funkce.

I přes svou oblíbenost, je metoda NM ve své základní podobě značně omezená dimenzí (množství proměnných) řešeného problému. Při větších dimenzích je značně neefektivní a často nedokonverguje. Proto je v této práci implementována modifikovaná metoda NM, která reflektuje velikost simplexu ve výpočtovém mechanismu, resp. především to, jak generuje a pracuje se vstupními parametry (obecný algoritmus metody zůstává zachován). Zdrojem pro realizaci této výpočetní metody byl vědecký článek [10] publikovanými autory: Fuchang Gao a Lixing Han, v roce 2012.

Vstupními parametry metody jsou: optimalizovaná funkce, výchozí parametry pro optimalizaci (vektor v_0), konvergenční odchylka výpočtu a maximální počet iterací, koeficienty pro metodu flexibilního simplexu.

Výstupními parametry metody jsou: optimalizované hodnoty (výstupní vektor vv), počet iterací, funkční hodnota (odchylka).

Metoda pracuje s tzv. simplexem. Simplex je množina $N + 1$ bodů v N -rozměrném prostoru, kde N je počet neznámých dané funkce. Nejhmotatelnější, a nejpřesnější představa simplexu je, jako „tabulky“, popř. matice neznámých, které vstupují do funkce. Kde každý řádek představuje jedno řešení funkce, resp. jeden bod simplexu. V jednorozměrném prostoru má simplex tvar úsečky, ve dvojrozměrném trojúhelníka, ve třírozměrném tetraedru atd. Základem metody je obměna bodů simplexu (neznámých funkce), za „lepší“ resp. takové, které mají nižší funkční hodnotu.

Nejprve je nutno vytvořit výchozí simplex. Volba velikosti počátečního simplexu je esenciální pro mechanismus výpočtu. Příliš velký simplex vede k pomalému prohledávání, naproti tomu příliš malý simplex, má větší šanci uváznout v lokálním minimu. Způsobů, jak prakticky generovat výchozí simplex je celá řada. Metoda výpočtu výchozího simplexu implementovaná v aplikaci vychází z dimenze řešeného problému, její rovnice je

$$S_{ij} = \overrightarrow{v0}_j + sc \cdot D_{ij}, \quad (3.23)$$

kde: S -matice simplexu, v_0 -výchozí vektor neznámých, sc - scalefactor, D - vstupní matice, i - příslušný řádek simplexu $i = 1, 2, \dots, N + 1$, j -příslušný sloupec simplexu $j = 1, 2, \dots, N$.

Matice D je jednotkovou maticí o velikosti $N \times N$, ke které je výpočtem přidán jeden řádek. Poté je tedy výsledná velikost matice $N + 1 \times N$. Poslední řádek D matice je spočten dle vzorce

$$D_{N+1j} = \frac{(1 - \sqrt{N+1})}{N}. \quad (3.24)$$

Jak již bylo řečeno, nevhodná velikost výchozího simplexu neblaze ovlivňuje rychlost výpočtu. Parametr scalefactor- sc má za úkol, dát výchozímu simplexu správný „rozměr“, tak aby následné iterace probíhali, pokud možno co nejrychleji. Hodnota sc vychází, z vektoru vstupních hodnot v_0 , nicméně nechceme, aby nabývala extrémních hodnot prakticky to znamená byla větší než 10, popř. menší než 1. Hodnoty sc se tedy volí dle následujícího klíče: Jako hodnotu sc vyber největší hodnotu z vektoru v_0 . Pokud některá z hodnot $v_{0j} > 10$, poté je $sc = 10$. Pokud není žádný z $v_{0j} > 1$, poté je $sc = 1$.

Dále je před započítáním samotného výpočtu nutno zvolit koeficienty reflexe α , expanze β , kontrakce γ a redukce δ . Značení koeficientů, je stejné, jako název proměnných, které reprezentují. Dle [10] jsou koeficienty voleny dle dimenze problému a doporučených konstant:

$$\alpha = c_1, \quad (3.25)$$

$$\beta = c_2 + \frac{2}{N+1}, \quad (3.26)$$

$$\gamma = c_3 - \frac{1}{N+1}, \quad (3.27)$$

$$\delta = c_4 - \frac{1}{N+1}, \quad (3.28)$$

kde: c_1 až c_4 jsou numerické konstanty.

Numerické konstanty c_1 až c_4 lze nastavit přímo v mobilní aplikaci, a tím změnit parametry výpočtu, nicméně dle [10], jsou nejvýhodnější hodnoty: $c_1 = 1$, $c_2 = 1$, $c_3 = 0,75$ a $c_4 = 1$.

Samotný algoritmus poté opakovaně provádí tyto kroky: reflexe, expanze, kontrakce, popř. redukce (záleží na hodnotách simplexu), dokud nejsou splněny ukončující podmínky metody.

3.4.1 Algoritmus metody NM

Každý iterační krok algoritmu NM je započat výpočtem funkčních hodnot simplexu, jejich seřazením, od nejlepšího bodu x_1 po nejhorší bod x_{N+1} , a výpočtem těžiště simplexu. Těžiště simplexu je počítáno, jako průměr vrcholů simplexu, s výjimkou nejhoršího bodu simplexu

$$x_0 = \frac{1}{n} \sum_{j=1}^n x_j, \quad (3.29)$$

kde: x_j -je bod simplexu na j -té pozici (výpočet probíhá po řádcích S_{ij}).

3.4.1.1 Reflexe

Poté následuje hledání nového bodu simplexu. Prvním bodem, který přichází v úvahu, je tzv reflexní bod (reflexní, jelikož pro jeho nalezení se využívá tzv. reflexe-zrcadlení nejhoršího bodu přes těžiště). Jedná se klíčovou hodnotu celé iterace. Reflexní bod je počítán dle následujícího vztahu

$$x_r = x_0 + \alpha(x_0 - x_{n+1}), \quad (3.30)$$

kde: x_0 -těžiště simplexu, x_{n+1} -nejhorší bod simplexu, α -reflexní koeficient.

Dále je testována podmínka

$$f(x_1) \leq f(x_r) < f(x_n), \quad (3.31)$$

kde: x_1 -nejlepší bod simplexu, x_r -bod reflexe; x_n - druhý nejhorší bod simplexu.

Pokud je tato podmínka splněna, je nejhorší bod simplexu nahrazen reflexním bodem, a celý algoritmus se opakuje.

3.4.1.2 Expanze

Pokud platí následující podmínka

$$f(x_r) < f(x_1), \quad (3.32)$$

kde: x_1 -nejlepší bod simplexu, x_r -bod reflexe.

Je počítán bod expanze dle následujícího vztahu

$$x_e = x_0 + \alpha \cdot \beta(x_0 - x_{n+1}), \quad (3.33)$$

kde: x_0 -těžiště simplexu; x_{n+1} -nejhorší bod simplexu, α -reflexní koeficient, β -expanzní koeficient.

Pokud platí následující podmínka:

$$f(x_e) < f(x_r), \quad (3.34)$$

kde: x_r -bod reflexe, x_e -bod expanze.

Dojde k nahrazení nejhoršího bodu simplexu bodem expanze. Pokud nikoliv, je nejhorší bod nahrazen bodem reflexe, a celý algoritmus se opakuje.

3.4.1.3 Kontrakce

Pokud nedošlo ke splnění podmínek reflexe (3.31) ani expanze (3.34), a tedy $f(x_r) > f(x_n)$. Jsou k dispozici dvě možnosti pokračování výpočtu:

a) Vnější kontrakce

Pokud platí následující podmínka:

$$f(x_r) < f(x_{n+1}), \quad (3.35)$$

kde: x_r -bod reflexe; x_{n+1} - nejhorší bod simplexu.

Je spočten bod vnější kontrakce dle vztahu:

$$x_{oc} = x_0 + \alpha \cdot \gamma(x_0 - x_{n+1}), \quad (3.36)$$

kde: x_{oc} -bod vnější kontrakce, x_0 -těžiště simplexu, x_{n+1} -nejhorší bod simplexu, α -reflexní koeficient, γ - koeficient kontrakce.

Pokud je splněna podmínka:

$$f(x_{oc}) < f(x_r), \quad (3.37)$$

je nejhorší bod simplexu nahrazen bodem vnější kontrakce. Pokud tato podmínka není splněna, je provedena redukce.

b) Vnitřní kontrakce

Pokud platí následující podmínka:

$$f(x_r) > f(x_{n+1}), \quad (3.38)$$

kde: x_r -bod reflexe; x_{n+1} - nejhorší bod simplexu.

Je spočten bod vnitřní kontrakce dle vztahu:

$$x_{ic} = x_0 + \gamma(x_0 - x_{n+1}), \quad (3.39)$$

kde: x_{ic} -bod vnitřní kontrakce, x_0 -těžiště simplexu, x_{n+1} -nejhorší bod simplexu, γ - koeficient kontrakce.

Pokud je splněna podmínka:

$$f(x_{ic}) < f(x_r), \quad (3.40)$$

je nejhorší bod simplexu nahrazen bodem vnitřní kontrakce. Pokud tato podmínka není splněna, je provedena redukce.

3.4.1.4 Redukce

Při redukci dojde k nahrazení všech bodů simplexu, kromě bodu nejlepšího, a to dle následujícího vztahu:

$$x_i = x_1 + \delta(x_i - x_1) \quad i \in \{2, \dots, n + 1\}, \quad (3.41)$$

kde x_i -bod simplexu na i -té pozici, x_1 -nejlepší bod simplexu; δ -redukční koeficient.

Nahrazením všech bodů v simplexu, až na bod nejlepší, dojde ke smrštění celého simplexu směrem k nejlepšímu bodu, následně je celý algoritmus opakován.

3.4.2 Ukončovací podmínky

Optimalizace prováděná výše popsaným algoritmem, může být ukončena dvěma základními podmínkami. První je ukončení, pokud je rozdíl funkčních hodnot nejlepšího a nejhoršího vrcholu simplexu, menší než předem stanovená odchylka. Jinými slovy, řešení už se nezlepšuje nad předem stanovenou hodnotu. Tato podmínka je vyjádřena vztahem

$$f(x_{n+1}) - f(x_1) \leq \xi, \quad (3.42)$$

kde: x_{n+1} -nejhorší vrchol simplexu, x_1 -nejlepší vrchol simplexu, ξ - konvergenční odchylka.

Druhou podmínkou je dosažení maximálního počtu iteračních kroků. Tato podmínka je vyjádřena následujícím vztahem

$$it < it_{max} \quad (3.43)$$

kde: it -aktuální počet iteračních kroků; it_{max} - maximální zadaný počet iteračních kroků.

3.5 INTEGRAČNÍ METODA RK4

3.5.1 RK4

Metody Runge-Kutta patří ke klasickým jednokrokovým metodám numerického řešení obecných diferenciálních rovnic (dále pouze ODR). Svými autory: Carle Rungem a Wilhelmem Kuttou, byla vytvořena již kolem roku 1900 (oba autoři metodu objevili nezávisle na sobě na přelomu století). Zdrojem informací pro metodu byl zdroj [9]. Obecně lze tvar explicitní metody Runge-Kutty zapsat následovně:

$$y_{n+1} = y_n + h \sum_{i=1}^p w_i k_i, \quad (3.44)$$

kde: y_{n+1} -vypočtený bod, y_n - výchozí bod výpočtu, h -velikost kroku, w_i - váhy koeficientů, p -řád polynomu funkce $y(t)$, k_i -koeficienty odhadů derivace

$$k_i = f \left(t + a_i h, y_n + h \sum_{j=1}^{i-1} \beta_{ij} k_j \right), \quad (3.45)$$

kde: t -aktuální čas (výchozí hodnota osy x , pro daný krok), a_i -váhy kroku, β_i -váhy koeficientů.

Ze všech metod Runge-Kutta, je metoda čtvrtého řádu (dále pouze RK4), v praxi nejvyužívanější a nejoblíbenější. Jelikož poskytuje dobrý kompromis, mezi přesností výsledků, náročností na výpočet, a složitostí implementace. Z těchto důvodů je i v této práci využito právě metody RK4.

Konkrétní podoba metody RK4, jež je v práci využita je

$$y_{i+1} = y_i + \frac{h}{6} (k_1 + 2k_2 + 2k_3 + k_4), \quad (3.46)$$

kde

$$k_1 = f(t_i, y_i), \quad (3.47)$$

$$k_2 = f \left(t_i + \frac{h}{2}, y_i + \frac{h}{2} k_1 \right), \quad (3.48)$$

$$k_3 = f \left(t_i + \frac{h}{2}, y_i + \frac{h}{2} k_2 \right), \quad (3.49)$$

$$k_4 = f(t_i + h, y_i + h k_3). \quad (3.50)$$

3.5.2 Převod ODR vyššího řádu na soustavu ODR prvního řádu

Metoda RK4, využít pouze pro ODR prvního řádu. Rovnice 3.3, kterou, je však nutno vyřešit, může (dle provedení aplikace), nebývat až řádu osmého. Tento problém lze prakticky vyřešit tak, že se ODR vyššího řádu rozloží na soustavu ODR prvního řádu, za pomoci substituce.

Použití metody snižování řádu derivace

$$a_n y^n(t) + a_{n-1} y^{n-1}(t) + \dots + a_1 y'(t) + a_0 y(t) = u, \quad (3.51)$$

substituce, na soustavu ODR prvního řádu:

$$y^n = x_n \rightarrow x_{n+1} = x'_n, \quad (3.52)$$

výslednou soustavu lze zapsat následovně

$$\begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_n \end{bmatrix} = \begin{bmatrix} 0 & 0 & \dots & 1 \\ 0 & \dots & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ a_1 & a_2 & \dots & a_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u, \quad (3.53)$$

Maticová rovnice soustavy vypadá následovně

$$X' = AX + B \cdot u, \quad (3.54)$$

kde: X' - matice derivací, A -matice koeficientů levé strany, X -matice proměnných levé strany, B -matice proměnných pravé strany.

Kterou lze již řešit pomocí metody RK4, přičemž s začíná rovnicí s nejnižším koeficientem, a substitucí spočtených koeficientů se postupně dořeší celá soustava. Kde řešení rovnice nejvyššího koeficientu, je řešením ODR rovnice vyššího řádu.

3.5.3 Implementace metody RK4

Jak již bylo zmíněno, integrační metoda má za úkol pracovat s rovnicí obecného dynamického systému 3.3. Celý postup integrace je tedy následující:

A) Rozdělit rovnici na dvě

Nejprve je nutno rovnici 3.3 rozdělit na dvě rovnice

$$a_n y^n(t) + a_{n-1} y^{n-1}(t) + \dots + a_1 y'(t) + a_0 y(t) = u, \quad (3.55)$$

$$y = b_m u^m(t) + b_{m-1} u^{m-1}(t) + \dots + b_1 u'(t) + b_0 u(t), \quad (3.56)$$

pokud aplikujeme Laplaceovu-transformaci, dostaneme rovnice (vztah vychází z obrazového přenosu)

$$a_n z^n + a_{n-1} z^{n-1} + \dots + a_1 z' + a_0 z = u, \quad (3.57)$$

$$y = b_m z^m + b_{m-1} z^{m-1} + \dots + b_1 z' + b_0 z, \quad (3.58)$$

přičemž platí rovnosti stran $P = L$, resp.

$$u = y, \quad (3.59)$$

a rovnost jednotlivých derivací z

$$z^n = z^m. \quad (3.60)$$

B) Spočítat řešení levé strany pro konstantu u

Dále je spočteno řešení levé strany rovnice (vztah 3.57), dle postupu snižování derivace vysvětleného v 3.5.2 a metody RK4 vysvětlené v 3.5.1. Přičemž za u je v případě implementace identifikace volena konstanta 1 reprezentující jednotkový skok a v případě implementace seřizování regulátoru hodnota regulační veličiny u . V tomto kroku jsou tedy získány hodnoty $z', z'' \dots z^n$.

C) Dořešit pravou stranu rovnice

Pro dořešení celé úlohy 3.3, zbývá dořešit pravou stranu rovnice 3.58. Toho je docíleno součtem jednotlivých členů rovnice 3.58 (kde koeficienty $b_1, b_2 \dots b_m$ jsou hodnoty navržené metodou NM a hodnoty jednotlivých derivací $z', z'' \dots z^m$, jsou získány z výpočtu levé strany rovnice). Takto spočtená hodnota y , je následně porovnávána s naměřenými daty (v případě identifikace), nebo zadanou hodnotou (v případě PID regulace). A následně z nich je spočtena odchylka pro metodu NM. Celý cyklus je opakován dle schémat obr. 3.2, obr. 3.13.

4 VERIFIKACE

Ověření výpočtového mechanismu aplikace bylo provedeno certifikovaným výpočtovým softwarem MATLAB ver. 2019 a jeho příslušnými zmíněnými moduly. Všechny verifikační výpočty byly provedeny na fyzickém zařízení: Xiaomi Redmi 7A.

Pro účely verifikace, bylo vybráno celkem šest ODR, představujících různé obrazové přenosy dynamických spojitéch systémů, s různými vlastnostmi, složitostí a řádem:

A) ODR prvního řádu levé strany a nultého řádu pravé strany:

$$3y' + 2y = 4u. \quad (4.1)$$

B) ODR druhého řádu levé strany a prvního řádu pravé strany:

$$y'' + 3y' + 2y = 2u' + 2u. \quad (4.2)$$

C) ODR třetího řádu levé strany a druhého řádu pravé strany:

$$2y''' + 3y'' + 2y' + 4y = 2u'' + 4u' + 8u. \quad (4.3)$$

D) ODR čtvrtého řádu levé strany a nultého řádu pravé strany:

$$y^{IV} + 5y''' + 8y'' + 4y' + 4y = u. \quad (4.4)$$

E) ODR pátého řádu levé strany a nultého řádu pravé strany:

$$2y^V + 12y^{IV} + 41y''' + 55y'' + 56y' + 8y = u. \quad (4.5)$$

F) ODR šestého řádu levé strany a nultého řádu pravé strany:

$$2y^{VI} + 12y^V + 41y^{IV} + 55y''' + 56y'' + 8y' + 5y = u. \quad (4.6)$$

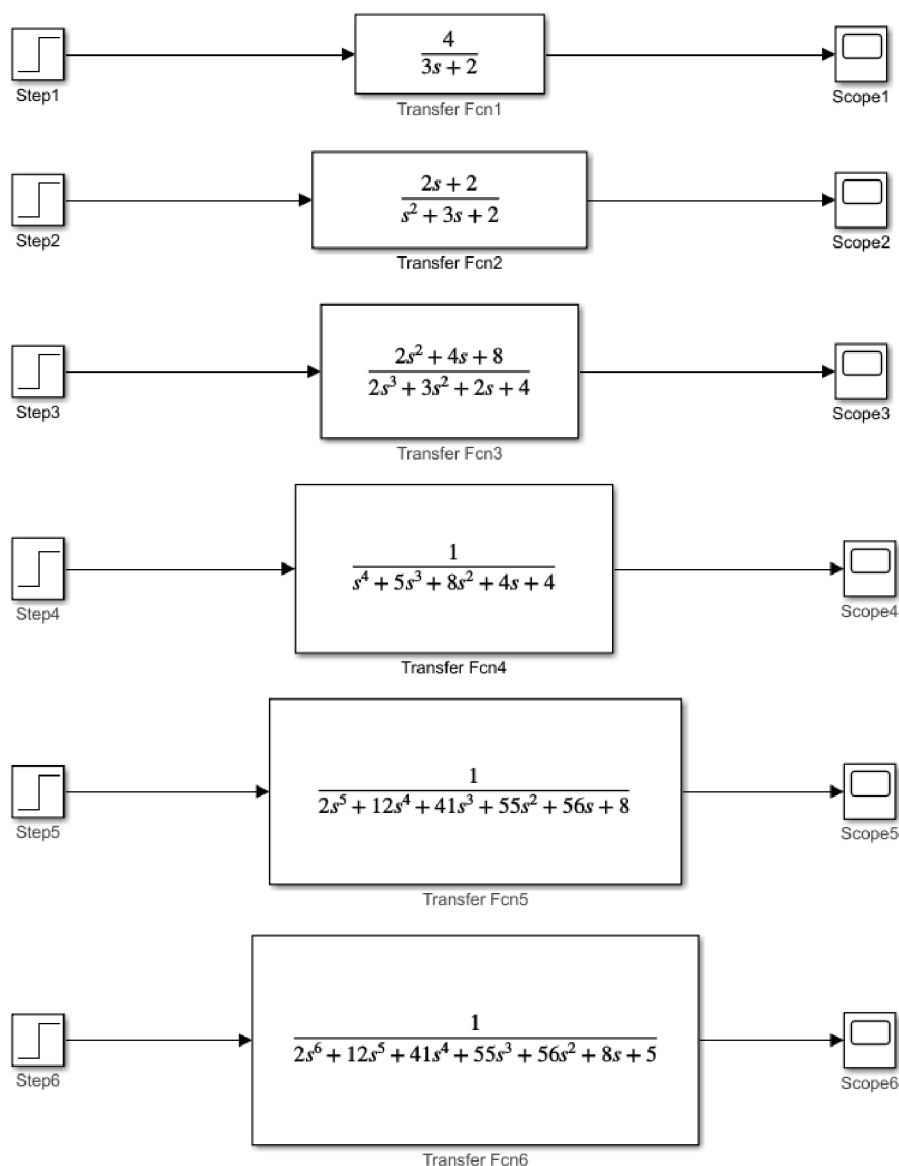
4.1 VERIFIKACE INTEGRACE (METODA RK4)

Výše zmíněné systémy byly vymodelovány v graficko-výpočetním prostředí MATLAB Simulink. Jak ukazuje obr. 4.1. Pomocí bloků Transfer Function a Step. Blok Scope slouží pro zobrazování a ukládání výsledků integrace. Hodnota funkce Step (skoku), je ve všech případech nastavena na jedna (jednotkový skok), s nulovým dopravním zpožděním. Časový horizont simulace je 0 až 10. Krok vzorkování je v blocích Scope nastavena na 0.05. Daný „Simulinkový“ model (obr. 4.1) představuje základní východisko, pro všechny další verifikační „Simulinkové“ modely, zmíněné v dalších kapitolách verifikace.

Takto nasimulovaná data byla poté převedena do formátu textového souboru (viz. 2.4), a nahrána do mobilního zařízení. Kde sloužila, jako „identifikační množina“ ale na samotný výpočet neměla vliv. Kde byla provedena integrace v mírně pozměněné aktivitě Identifikace (byla vyřazena metoda optimalizace, a za integrační koeficienty byly přímo dosazeny koeficienty zkušebních ODR). Aby byla výchozí data pro výpočet integrace v obou případech totožná. Fakt, že se graf vykreslený externě

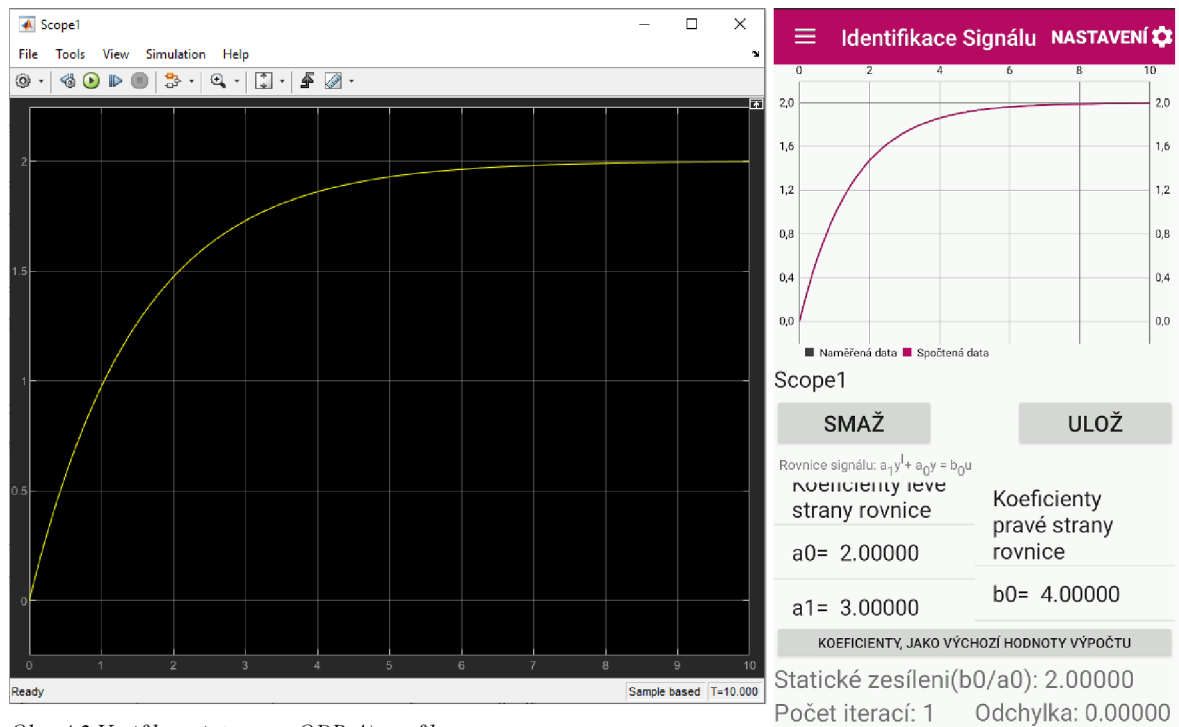
nahránímí daty výsledku simulace v Matlab Simulink, a graf spočten z koeficientů zkušebního modelu překrývají je jen další vrstvou pro ověření správné funkčnosti integračního aparátů.

Grafické a numerické porovnání těchto metod integrace je zobrazeno na obrázcích 4.2 až 4.13 (vstupní soubor dat z MATLAB Simulink vlevo (převedené do formátu .txt pro import dat do mobilní aplikace), výpis výsledků integrace v consoli Android Studio vpravo). Jak je vidět data jsou prakticky totožná. Mírná neshoda se vyskytuje až oblasti pátého desetinného řádu výpočtu. Její příčinou je buďto mírný rozdíl použitých metod integrace, nebo mírné odlišnosti v koeficientech výpočetních integračních metod. Daná přesnost výpočtu je pro danou aplikaci naprosto dostačující.



Obr. 4.1 MATLAB Simulink model obrazových přenosů pro kontrolu metody integrace

A) ODR prvního řádu levé strany a nultého řádu pravé strany:



Obr. 4.2 Verifikace integrace ODR A) grafika

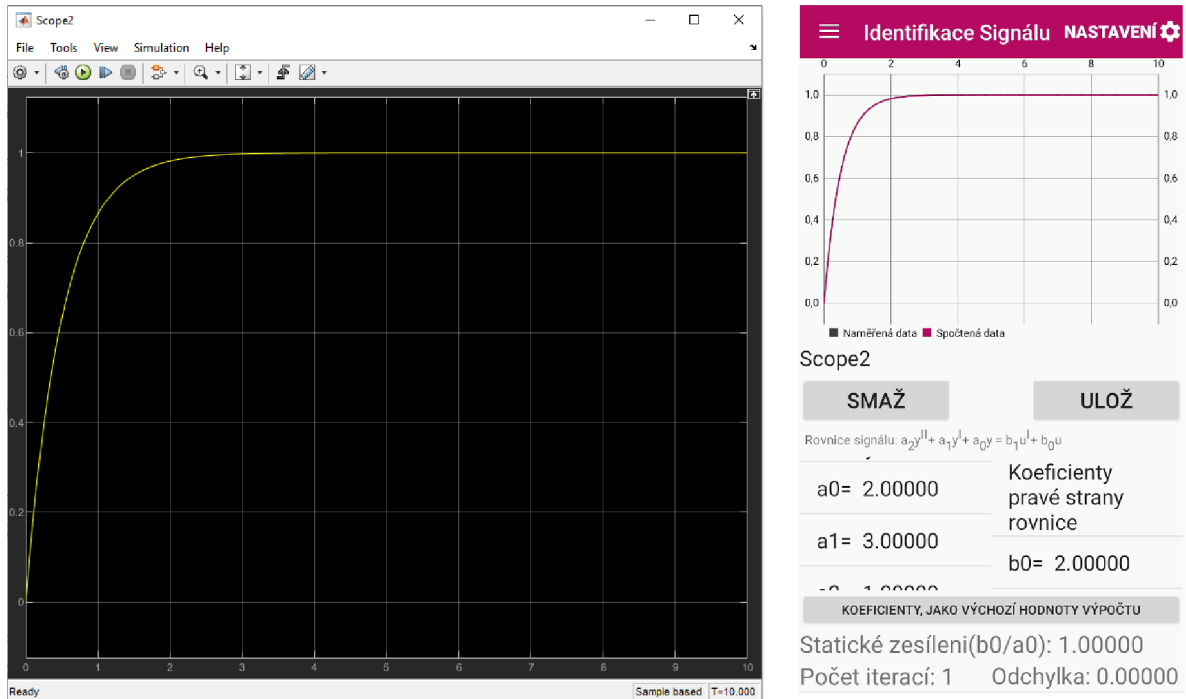
Scope1 - Poznámkový blok

Soubor	Úpravy	Formát	Zobrazení	Nápověda
0.000000e+00	0.000000e+00			
5.000000e-02	6.5567799e-02			
1.000000e-01	1.2898603e-01			
1.500000e-01	1.9032516e-01			
2.000000e-01	2.4965336e-01			
2.500000e-01	3.0703655e-01			
3.000000e-01	3.6253849e-01			
3.500000e-01	4.1622087e-01			
4.000000e-01	4.6814332e-01			
4.500000e-01	5.1836356e-01			
5.000000e-01	5.6693738e-01			
5.500000e-01	6.1391876e-01			
6.000000e-01	6.5935991e-01			
6.500000e-01	7.0331132e-01			
7.000000e-01	7.4582183e-01			
7.500000e-01	7.8693868e-01			
8.000000e-01	8.2670756e-01			
8.500000e-01	8.6517266e-01			
9.000000e-01	9.0237673e-01			
9.500000e-01	9.3836110e-01			
1.000000e+00	9.7316576e-01			
1.050000e+00	1.0068294e+00			
1.100000e+00	1.0393894e+00			
1.150000e+00	1.0708820e+00			
1.200000e+00	1.1013421e+00			
1.250000e+00	1.1308036e+00			
1.300000e+00	1.1592992e+00			
1.350000e+00	1.1868607e+00			
1.400000e+00	1.2135186e+00			
1.450000e+00	1.2393025e+00			
1.500000e+00	1.2642411e+00			
1.550000e+00	1.2883622e+00			
1.600000e+00	1.3116924e+00			
1.650000e+00	1.3342578e+00			
1.700000e+00	1.3560835e+00			
1.750000e+00	1.3771936e+00			
1.800000e+00	1.3976116e+00			

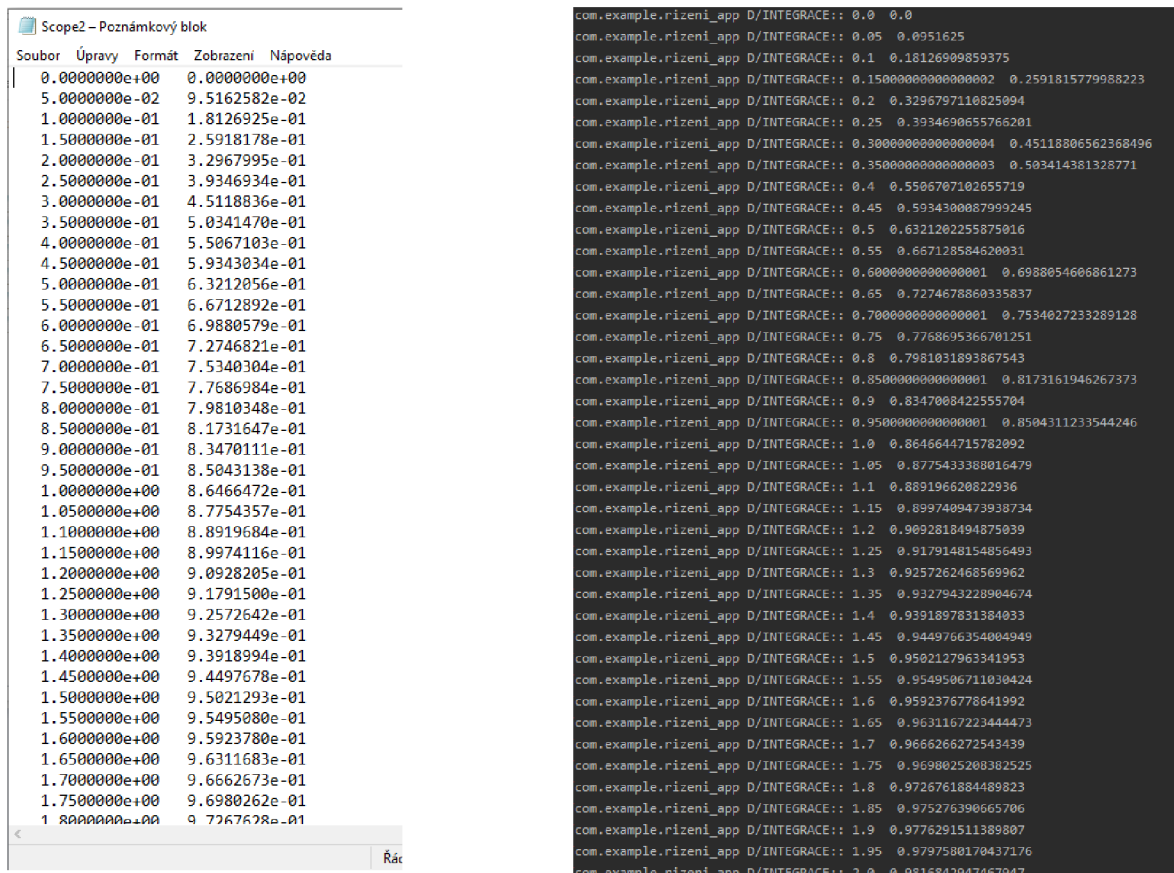
```
com.example.rizeni_app D/INTEGRACE:: 0.0 0.0
com.example.rizeni_app D/INTEGRACE:: 0.05 0.06556779835390947
com.example.rizeni_app D/INTEGRACE:: 0.1 0.12898602861732947
com.example.rizeni_app D/INTEGRACE:: 0.15000000000000002 0.1903251620138126
com.example.rizeni_app D/INTEGRACE:: 0.2 0.24965335944542366
com.example.rizeni_app D/INTEGRACE:: 0.25 0.30703654723408635
com.example.rizeni_app D/INTEGRACE:: 0.30000000000000004 0.3625384903798332
com.example.rizeni_app D/INTEGRACE:: 0.35000000000000003 0.41622086341736486
com.example.rizeni_app D/INTEGRACE:: 0.4 0.4681433189496544
com.example.rizeni_app D/INTEGRACE:: 0.45 0.5183635539347534
com.example.rizeni_app D/INTEGRACE:: 0.5 0.566937373799458
com.example.rizeni_app D/INTEGRACE:: 0.55 0.6139187544510785
com.example.rizeni_app D/INTEGRACE:: 0.60000000000000001 0.6593599022562221
com.example.rizeni_app D/INTEGRACE:: 0.65 0.7033113120532368
com.example.rizeni_app D/INTEGRACE:: 0.70000000000000001 0.7458218232627812
com.example.rizeni_app D/INTEGRACE:: 0.75 0.7869386741588711
com.example.rizeni_app D/INTEGRACE:: 0.8 0.826707543607096
com.example.rizeni_app D/INTEGRACE:: 0.85000000000000001 0.865172655036308
com.example.rizeni_app D/INTEGRACE:: 0.9 0.9023767208455726
com.example.rizeni_app D/INTEGRACE:: 0.95000000000000001 0.9383610917636498
com.example.rizeni_app D/INTEGRACE:: 1.0 0.9731657546936026
com.example.rizeni_app D/INTEGRACE:: 1.05 1.006829385063172
com.example.rizeni_app D/INTEGRACE:: 1.1 1.039389390368775
com.example.rizeni_app D/INTEGRACE:: 1.15 1.070881951743238
com.example.rizeni_app D/INTEGRACE:: 1.2 1.1013420641607765
com.example.rizeni_app D/INTEGRACE:: 1.25 1.1308035753238999
com.example.rizeni_app D/INTEGRACE:: 1.3 1.1592992232754507
com.example.rizeni_app D/INTEGRACE:: 1.35 1.186860672777576
com.example.rizeni_app D/INTEGRACE:: 1.4 1.2135185504980526
com.example.rizeni_app D/INTEGRACE:: 1.45 1.2393024790430696
com.example.rizeni_app D/INTEGRACE:: 1.5 1.264241109874281
com.example.rizeni_app D/INTEGRACE:: 1.55 1.2883621551467106
com.example.rizeni_app D/INTEGRACE:: 1.6 1.3116924185028862
com.example.rizeni_app D/INTEGRACE:: 1.65 1.334257824857421
com.example.rizeni_app D/INTEGRACE:: 1.7 1.356083449205142
com.example.rizeni_app D/INTEGRACE:: 1.75 1.377193544484773
com.example.rizeni_app D/INTEGRACE:: 1.8 1.3976115685291406
com.example.rizeni_app D/INTEGRACE:: 1.85 1.4173602101318452
com.example.rizeni_app D/INTEGRACE:: 1.9 1.4364614142593648
com.example.rizeni_app D/INTEGRACE:: 1.95 1.4549364064366095
com.example.rizeni_app D/INTEGRACE:: 2.0 1.4728957163330204
```

Obr. 4.3 Verifikace integrace A) numerika

B) ODR druhého řádu levé strany a prvního řádu pravé strany:

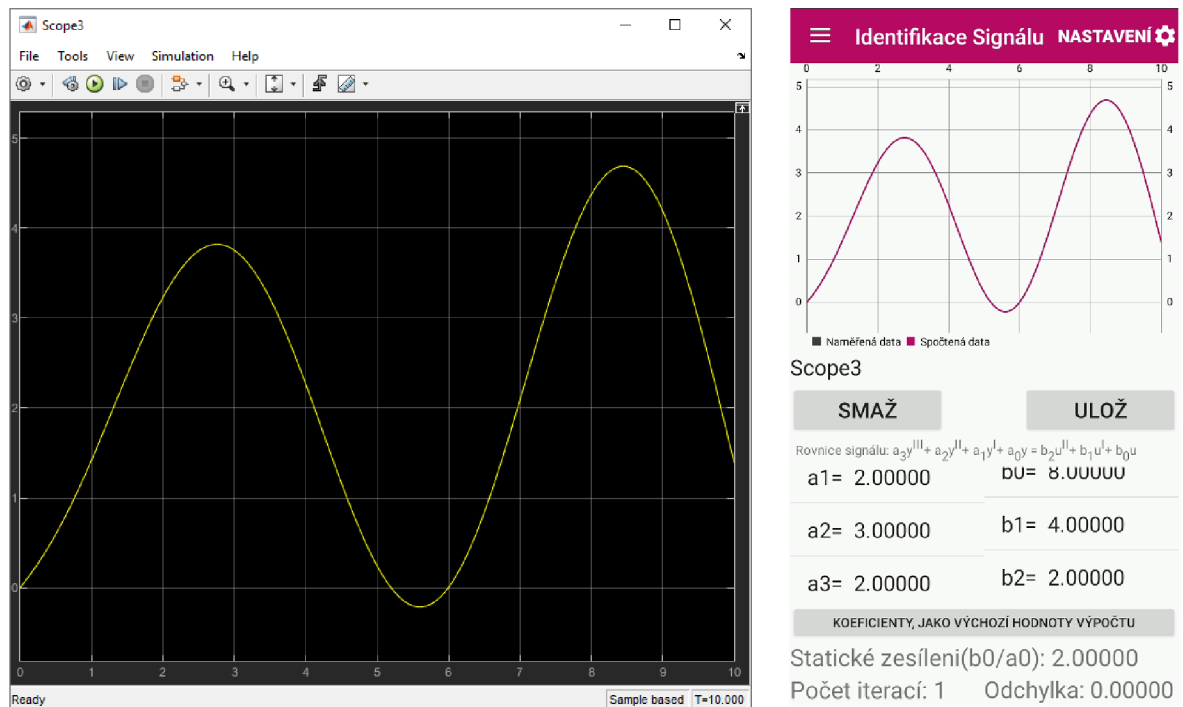


Obr. 4.4 Verifikace integrace B) grafika

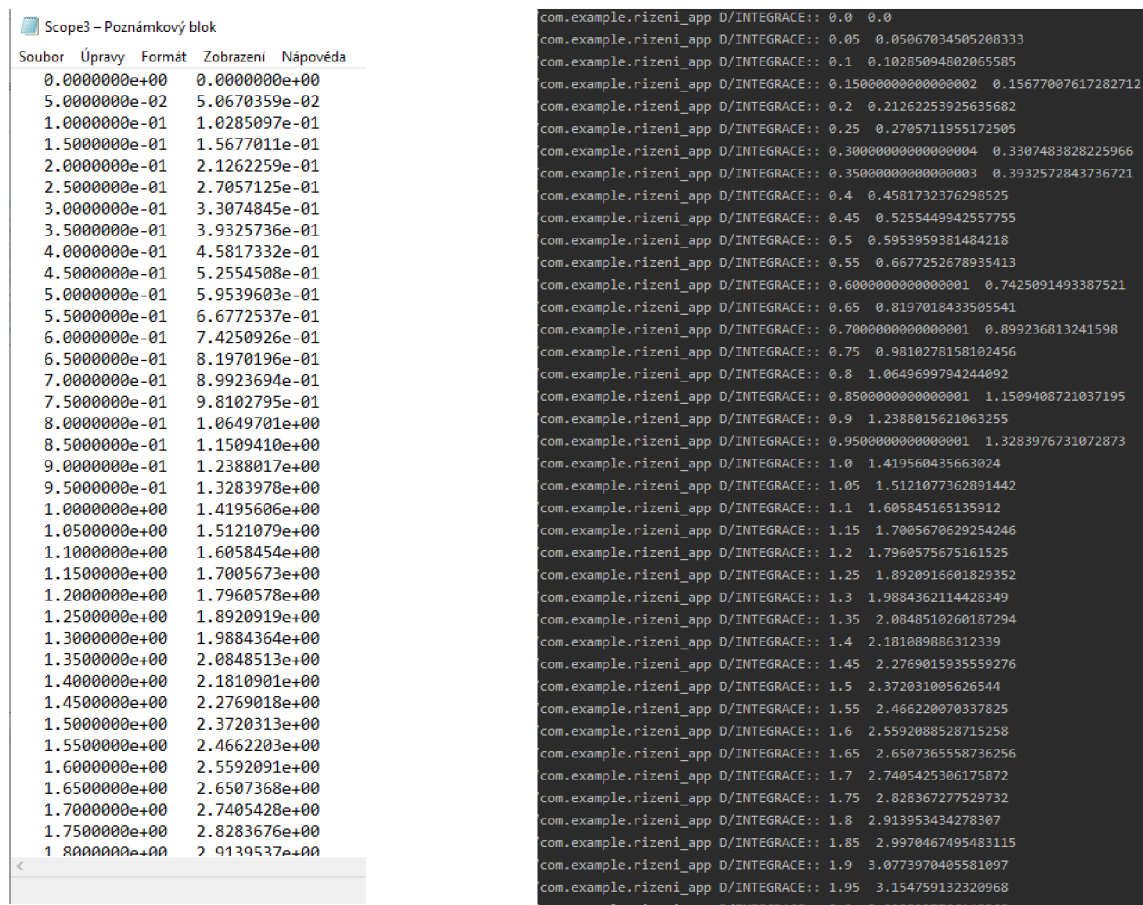


Obr. 4.5 Verifikace integrace B) numerika

C) ODR třetího řádu levé strany a druhého řádu pravé strany:

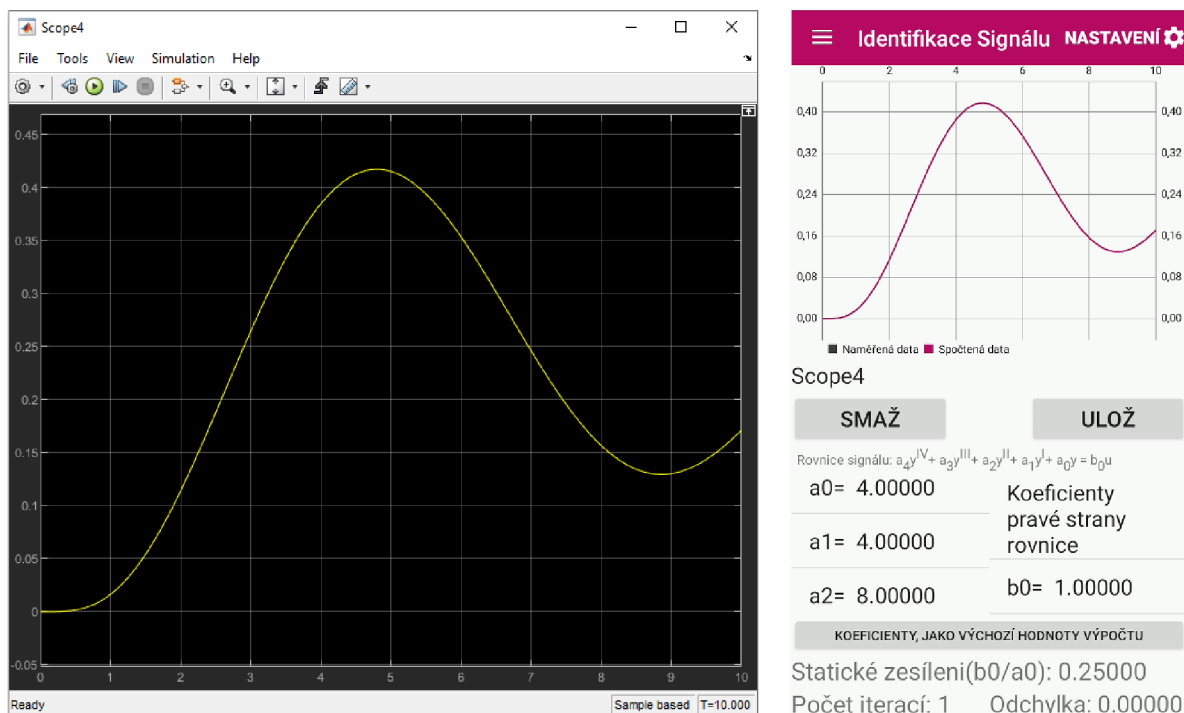


Obr. 4.6 Verifikace integrace C) grafika

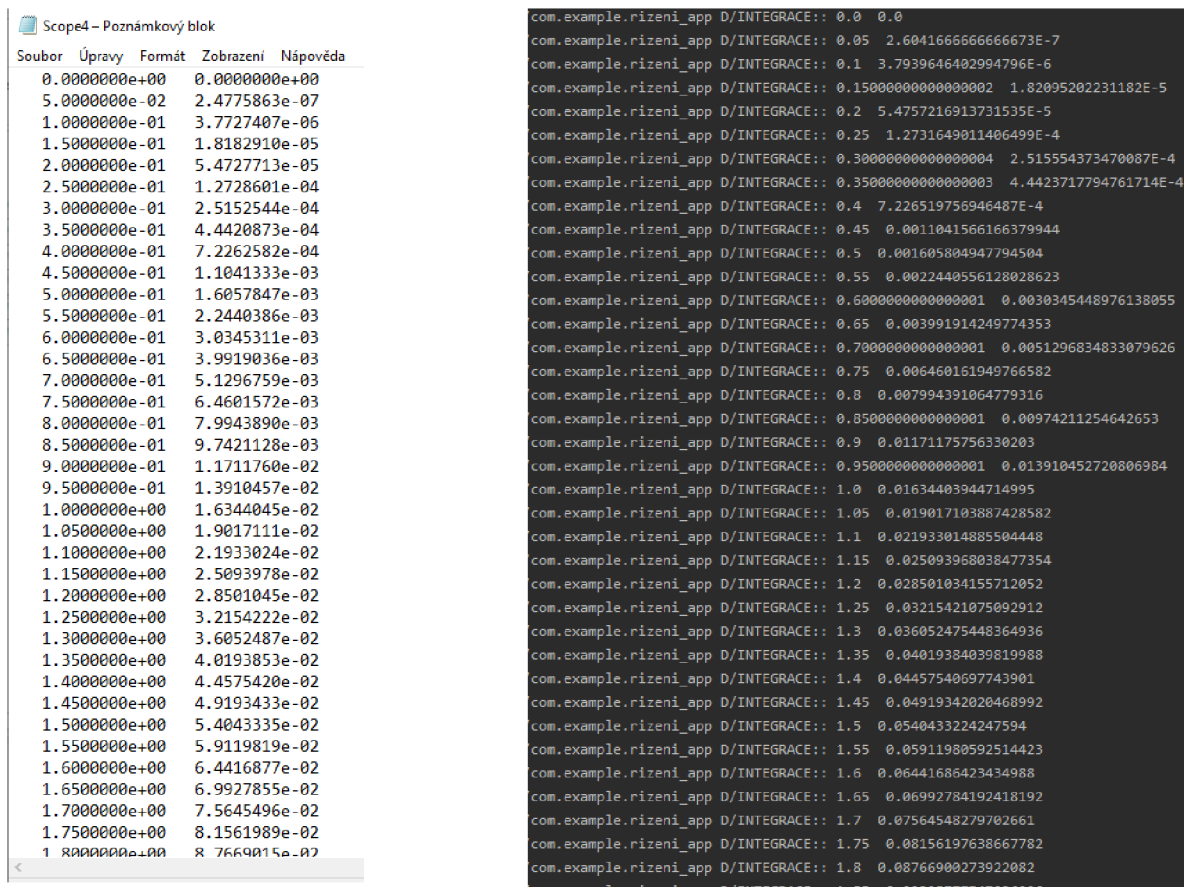


Obr. 4.7 Verifikace integrace C) numerika

D) ODR čtvrtého řádu levé strany a nultého řádu pravé strany:

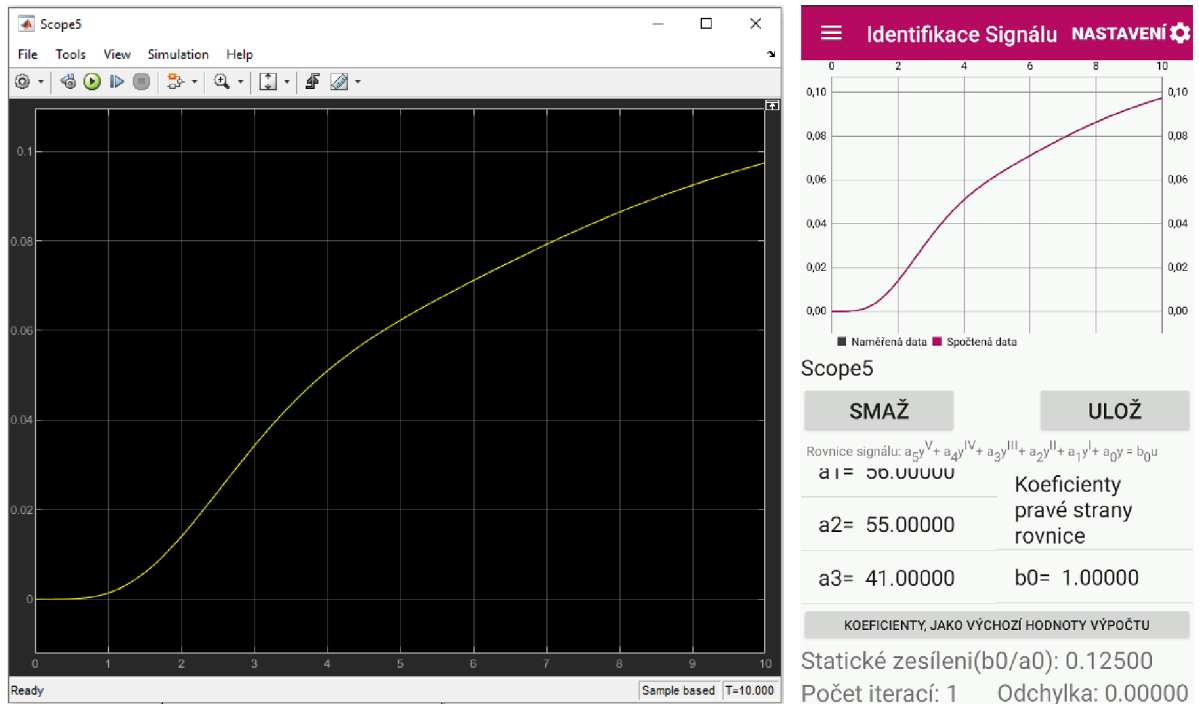


Obr. 4.8 Verifikace integrace D) grafika

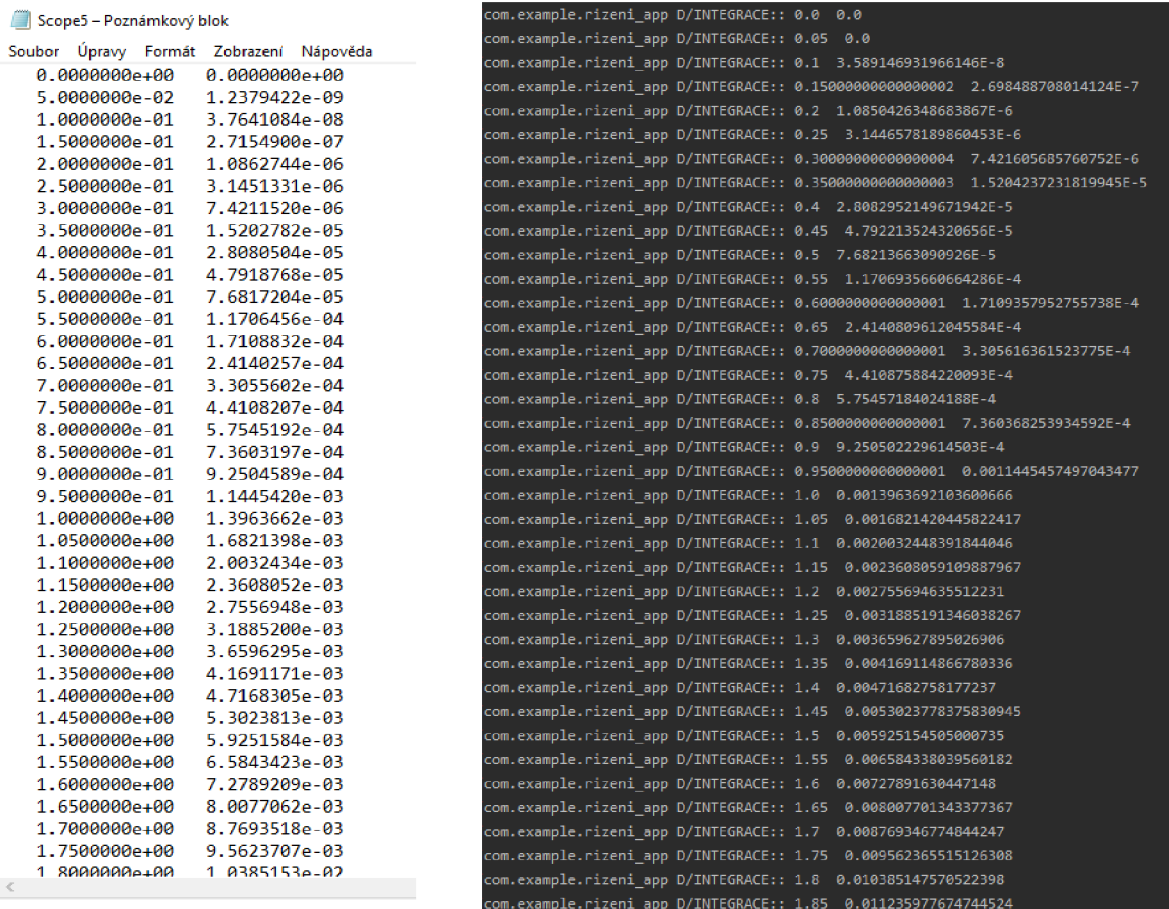


Obr. 4.9 Verifikace integrace D) numerika

E) ODR pátého řádu levé strany a nultého řádu pravé strany:

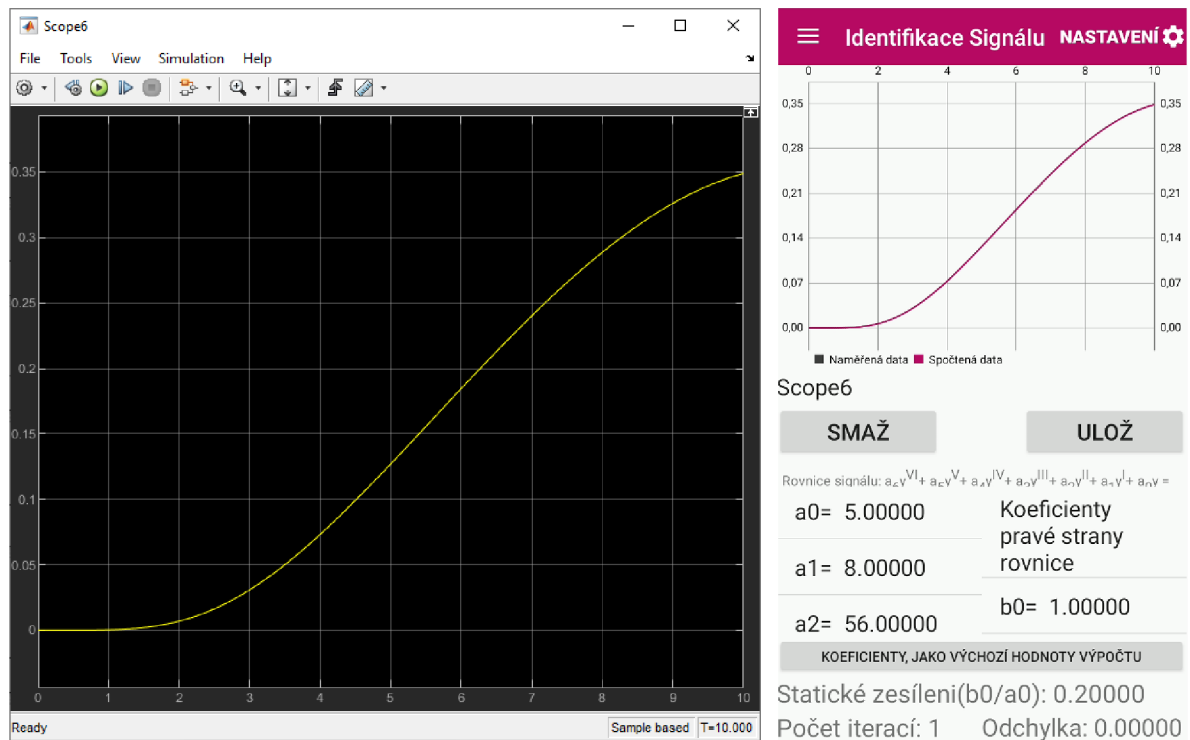


Obr. 4.10 Verifikace integrace E) grafika

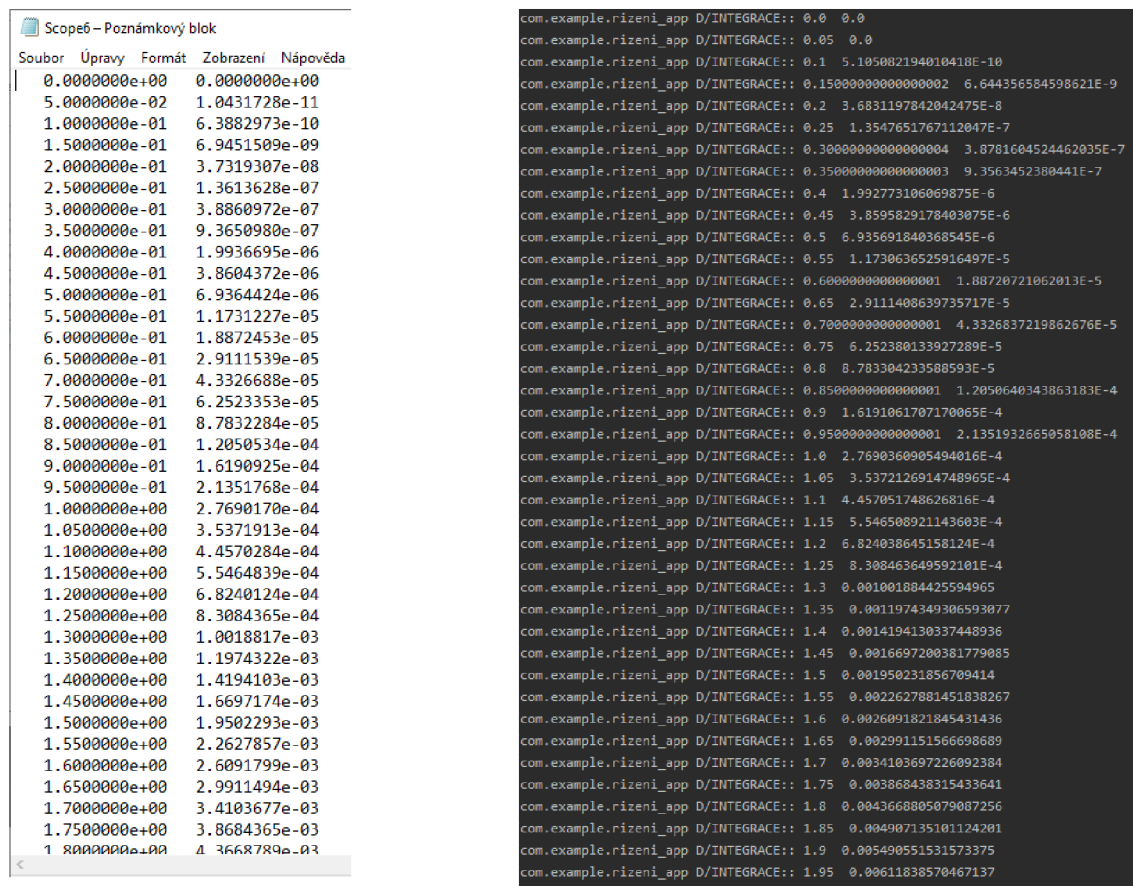


Obr. 4.11 Verifikace integrace E) numerika

F) ODR šestého řádu levé strany a nultého řádu pravé strany:



Obr. 4.12 Verifikace integrace F grafika



Obr. 4.13 Verifikace integrace F) numerika

4.2 VERIFIKACE IDENTIFIKACE (METODA NELDER-MEAD)

Pro verifikaci metody identifikace/optimalizace, je rozhodující ověření, její schopnosti co nejpřesněji „zrekonstruovat“ matematický model dynamického systému (najít správné koeficienty), z hodnot, jejichž skutečná hodnota byla „zdeformována“ rušením označovaným technickým termínem šum.

Šum se může skládat celkem ze tří složek:

- Vysokofrekvenční stochastická složka (Bílý šum)
- Nízkofrekvenční stochastická složka
- Šum neznámého původu

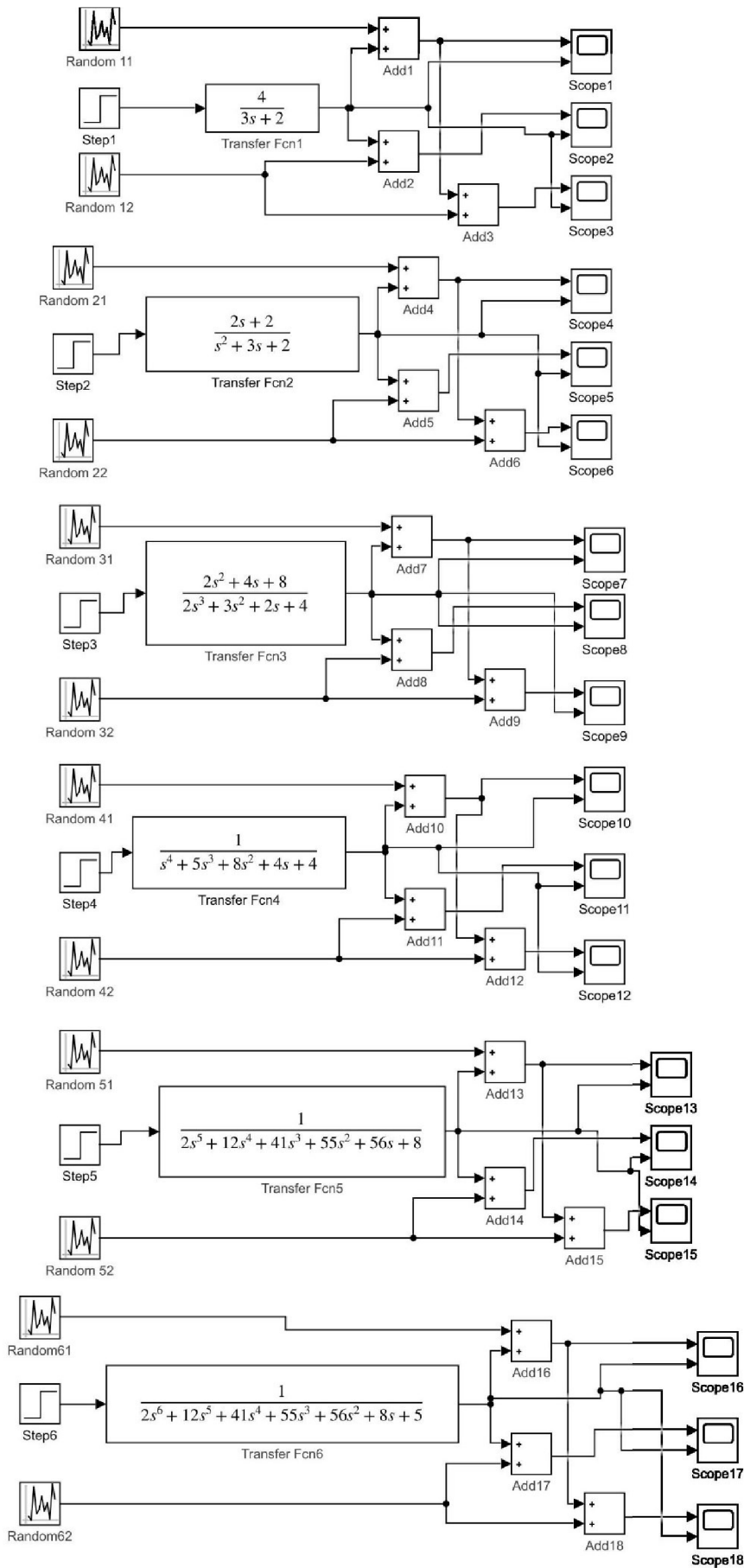
Skutečný praktický rozdíl mezi nízkofrekvenčním a vysokofrekvenčním šumem, je vzorkovací perioda měření. Nízkofrekvenční i vysokofrekvenční šum jsou stochastického (náhodného) charakteru, a způsobují, že hodnoty měřeného signálu oscilují, kolem jejich skutečné hodnoty. A to v určité toleranci výchylek, charakter těchto výchylek je naprosto náhodný s normálním rozdělením (obsahuje kladné i záporné přírůstky, a tedy při normálním rozdělení se suma všech těchto odchylek v nekonečnu rovná nule), takže celková tendence průběhu změřených dat je zachována.

S vysokofrekvenčním šumem, si umí poradit většina identifikačních metod. Nízkofrekvenční šum představuje již větší problém, nicméně optimalizační metoda implementovaná v této práci, se dokáže s tímto druhem šumu rovněž obstojně vypořádat, viz. níže. Poslední druh šumu (šum neznámého původu), představuje největší problém, jelikož je způsoben dalším dynamickým systémem, jako takový mění tendenci průběhu dat, a proto si s ním zde využívaná metoda identifikace neumí poradit. Problematika jeho kompenzace přesahuje rámec této práce a dále není v této práci rozebírána.

4.2.1 Generování verifikačních signálů

Pro generování signálů k účelům verifikace optimalizace, je využit model MATLAB Simulink (obr. 4.14), popsán v předchozí kapitole, ke kterému byly přidány generátory náhodných čísel s normálním rozdělením (Uniform Random Number), pro simulaci nízkého a vysokofrekvenčního šumu, a součtové členy (Add) pro kombinaci těchto signálů.

Nastavení bloků je následující: vzorkovací frekvence, pro sběr dat je nastavena na 0.05, vzorkovací frekvence generátoru náhodných čísel vysokofrekvenčního šumu je 0.01, vzorkovací frekvence generátoru nízkofrekvenčního šumu je 0.5. Rozmezí hodnot pro generátor náhodných čísel vysokofrekvenčního šumu je ± 0.05 . Rozmezí hodnot pro generátor náhodných čísel nízkofrekvenčního šumu je ± 0.1 . Všechna ostatní nastavení zůstávají stejná jako v modelu popsaném obrázkem 4.1.



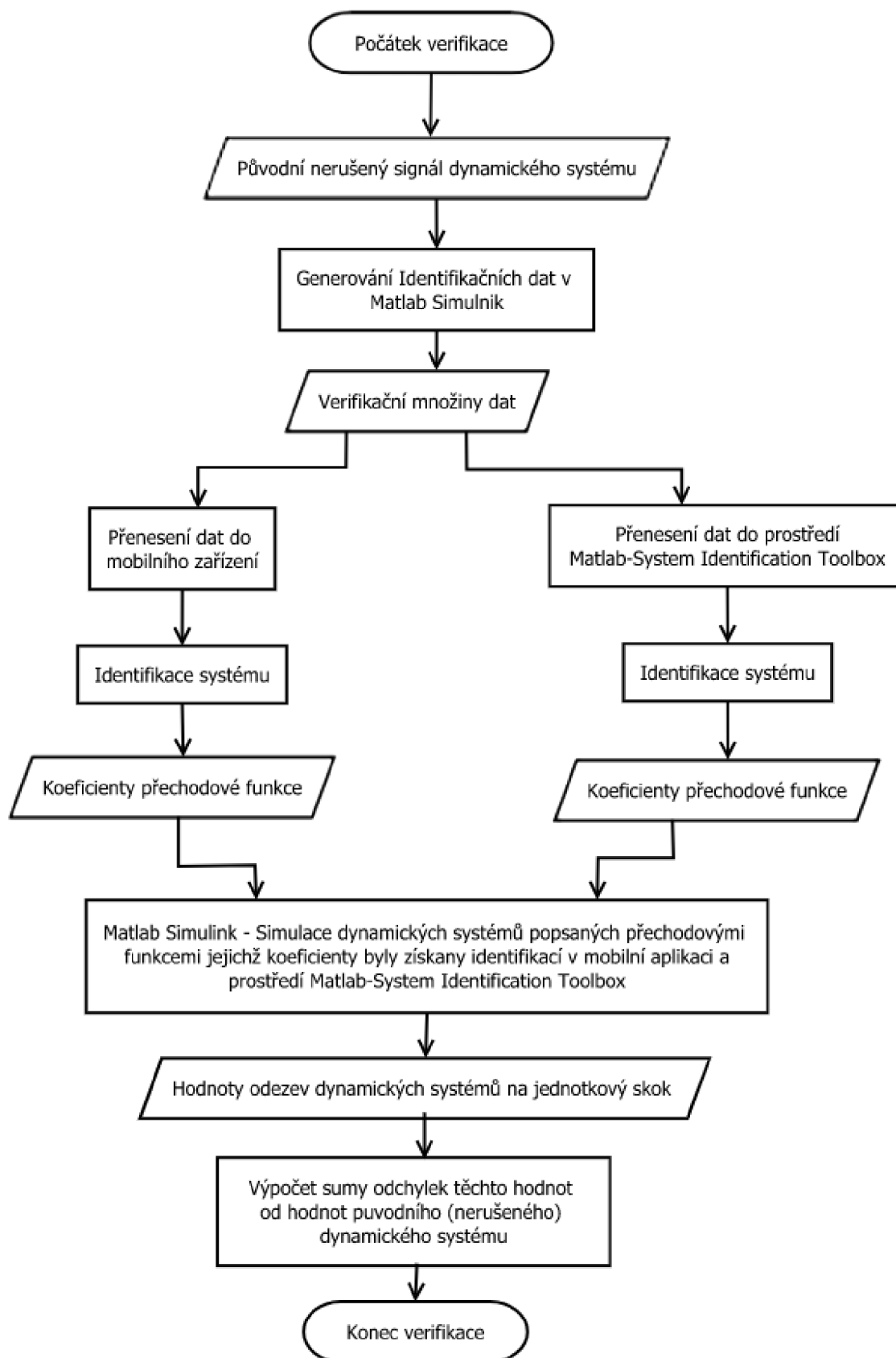
Obr. 4.14 MATLAB Simulink model pro generování verifikačních signálů identifikace dynamických systémů

4.2.2 Vyhodnocení identifikace verifikačních signálů

Jelikož porovnávání hodnot simplexů, s metodou `fminsearch` (simplexová metoda optimalizace nativní pro software MATLAB), neposkytuje dostatečně validní a názorná data (příliš mnoho simplexových hodnot (jejichž vypisování na papírový formát není efektivní), a dílčí hodnoty simplexu si v mnohých případech zcela neodpovídají, neboť je využívána speciálně upravená metoda svažujícího se simplexu, pro reflektování dimenze problému-viz. 3.4).

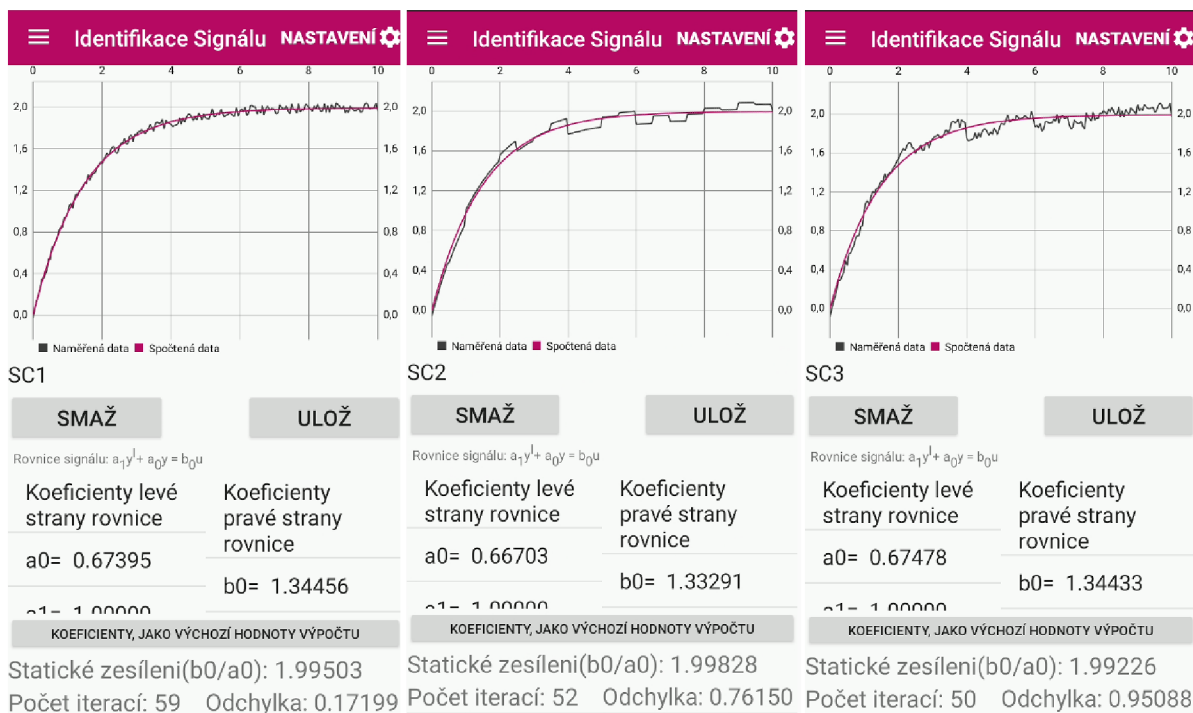
Je verifikace provedena formou srovnání simulace hodnot původního obrazového přenosu (synonymum je přenosová funkce anglicky „transfer function“) s obrazovým přenosem, do kterého jsou vloženy koeficienty spočtené aplikací, na základě dat se šumem. A následně je pro numerické porovnání stanovena suma absolutní odchylky těchto hodnot.

Pro účely dalšího srovnání se „správně vypočtenými“ koeficienty, obrazového přenosu, jsou data se šumem rovněž vyhodnocena pomocí vestavěného modulu softwaru MATLAB: Matlab-Systém Identification Toolbox. Schematicky je proces verifikace popsán obrázkem 4.15. Značení jednotlivých signálů a systémů, vychází z obr. 4.14. Identifikace pomocí Matlab-Systém Identification Toolbox, je prováděn modelem přechodové funkce příslušného řádu, jelikož tento model se nejvíce blíží výpočtovému modelu použitému v mobilní aplikaci. Je nutné si uvědomit, že výsledky z Matlab-Systém Identification Toolbox, jsou pouze orientační, neboť je v tomto případě využíván k poněkud jinému účelu, než ke kterému byl navrhnut, a navíc obsahuje několik dalších možností výpočtu, které by si s danými úlohami dokázali poradit mnohem lépe než zde prezentované výsledky.

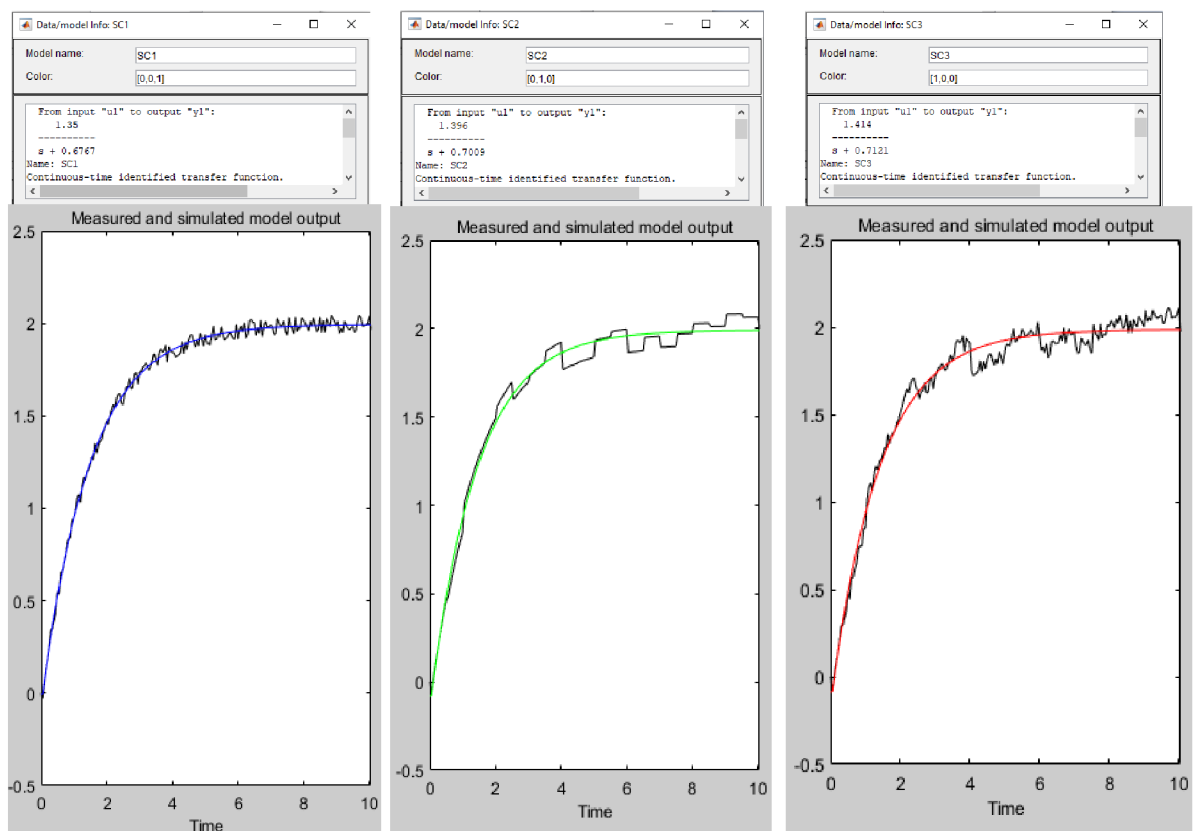


Obr. 4.15 Diagram postupu verifikace identifikace/optimalizace dynamického systému

A) ODR prvního řádu levé strany a nultého řádu pravé strany:



Obr. 4.16 Výsledek identifikace verifikačních signálů A) mobilní aplikací (zleva: vysokofrekvenční šum, nízkofrekvenční šum, kombinace vysokofrekvenčního a nízkofrekvenčního šumu)



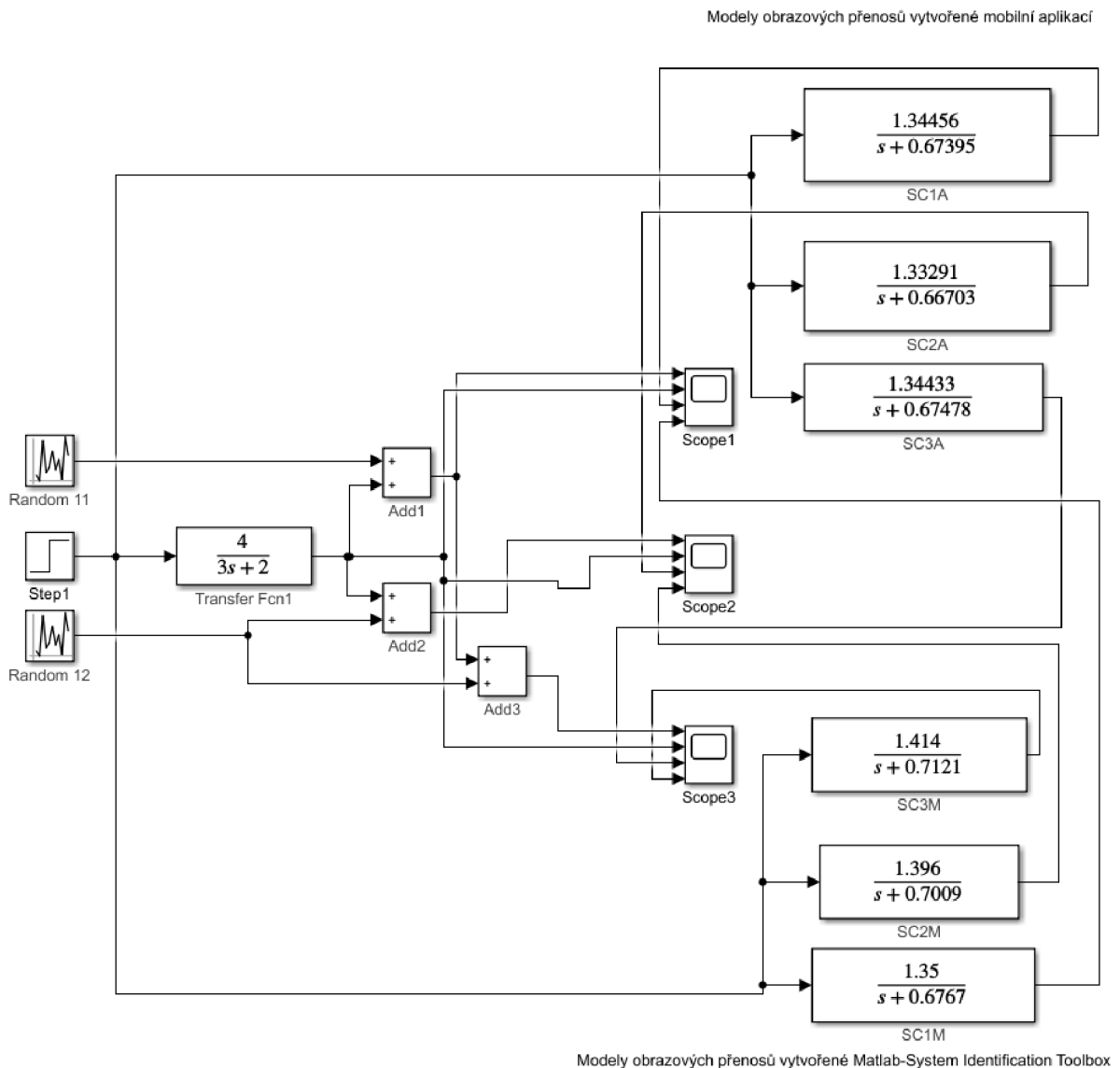
Obr. 4.17 Výsledek identifikace verifikačních signálů A) Matlab - System Identification Toolbox (zleva: vysokofrekvenční šum, nízkofrekvenční šum, kombinace vysokofrekvenčního a nízkofrekvenčního šumu)

Výsledky identifikace:

	MATLAB-System Identification Toolbox	Mobilní aplikace pro identifikaci a řízení signálů
SC1	$G(s) = \frac{1.35}{s + 0.6767}$	$G(s) = \frac{1.34456}{s + 0.67395}$
SC2	$G(s) = \frac{1.396}{s + 0.7009}$	$G(s) = \frac{1.33291}{s + 0.66703}$
SC3	$G(s) = \frac{1.414}{s + 0.7121}$	$G(s) = \frac{1.34433}{s + 0.67478}$

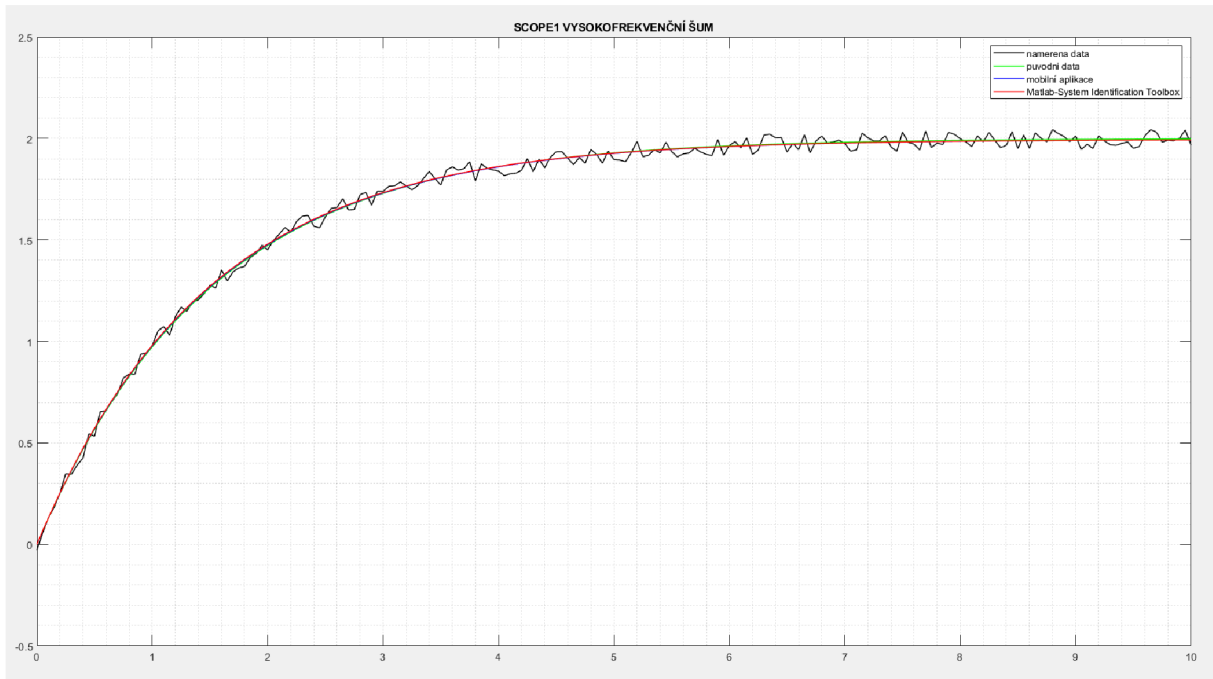
Tab. 4.1 Porovnání výsledků identifikace A)

Verifikační schéma Matlab Simulink:

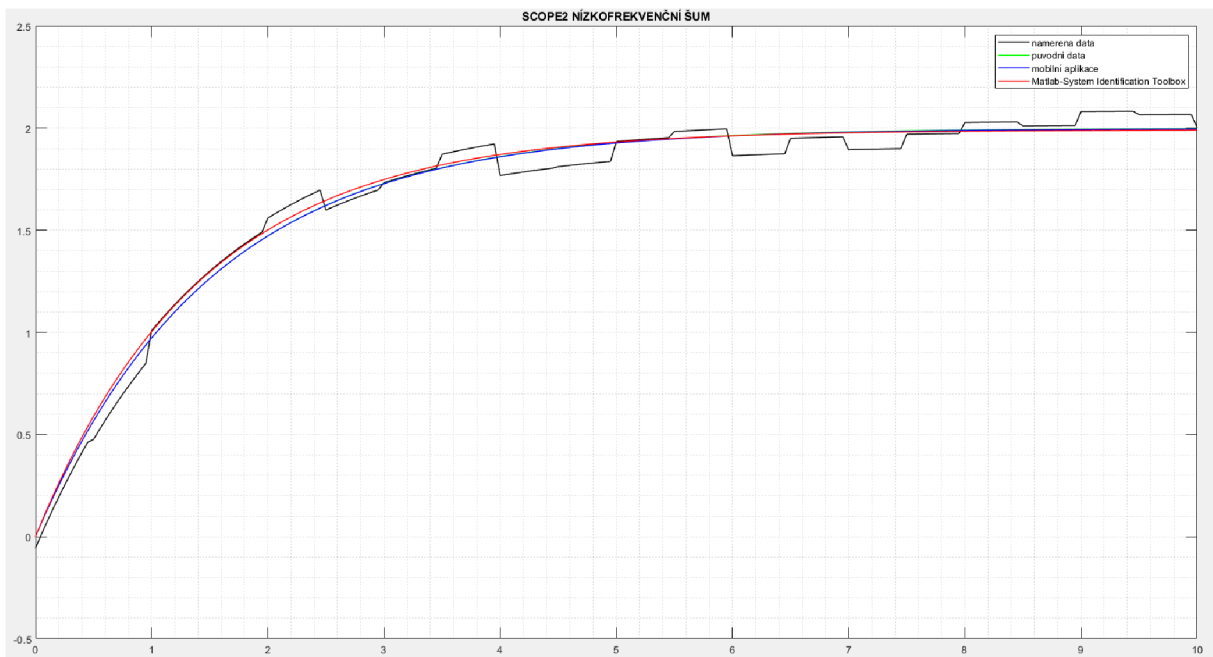


Obr. 4.18 Verifikační schéma A) identifikace/optimalizace Matlab Simulink

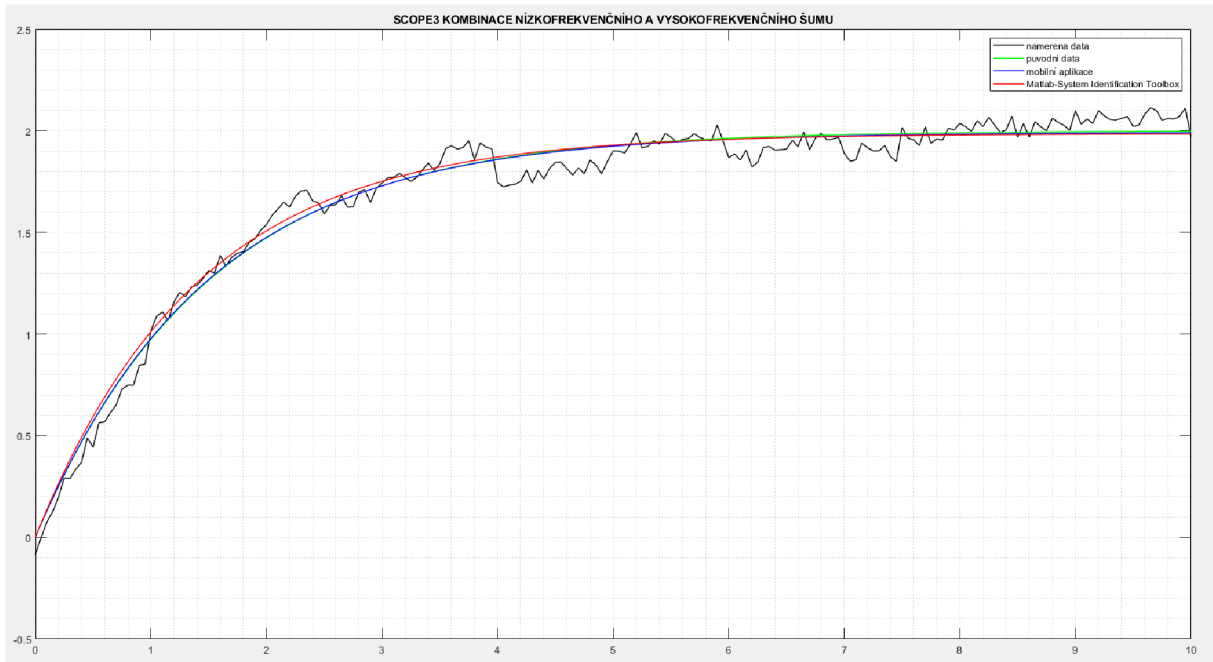
Grafické zobrazení výsledků verifikace identifikace:



Obr. 4.19 Grafické zobrazení výsledků verifikace identifikace A) vysokofrekvenční šum



Obr. 4.20 Grafické zobrazení výsledků verifikace identifikace A) nízkofrekvenční šum



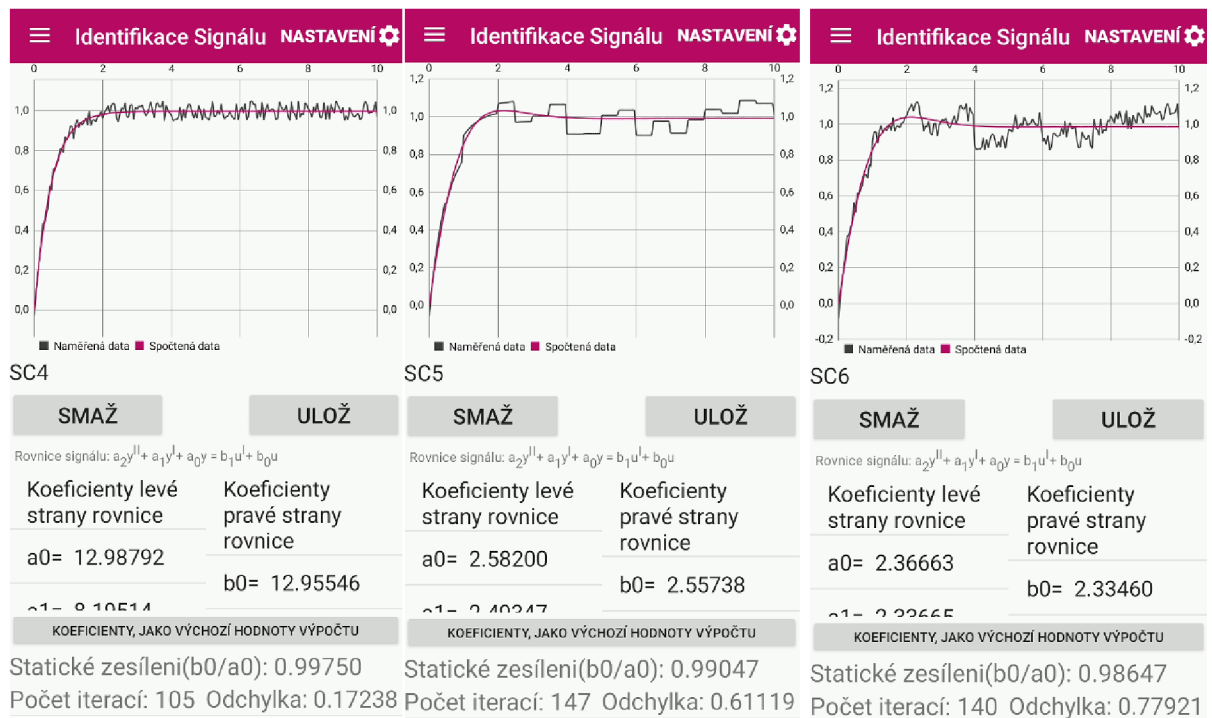
Obr. 4.21 Grafické zobrazení výsledků verifikace identifikace A) kombinace vysokofrekvenčního a nízkofrekvenčního šumu

Suma absolutních odchylek, od původního signálu:

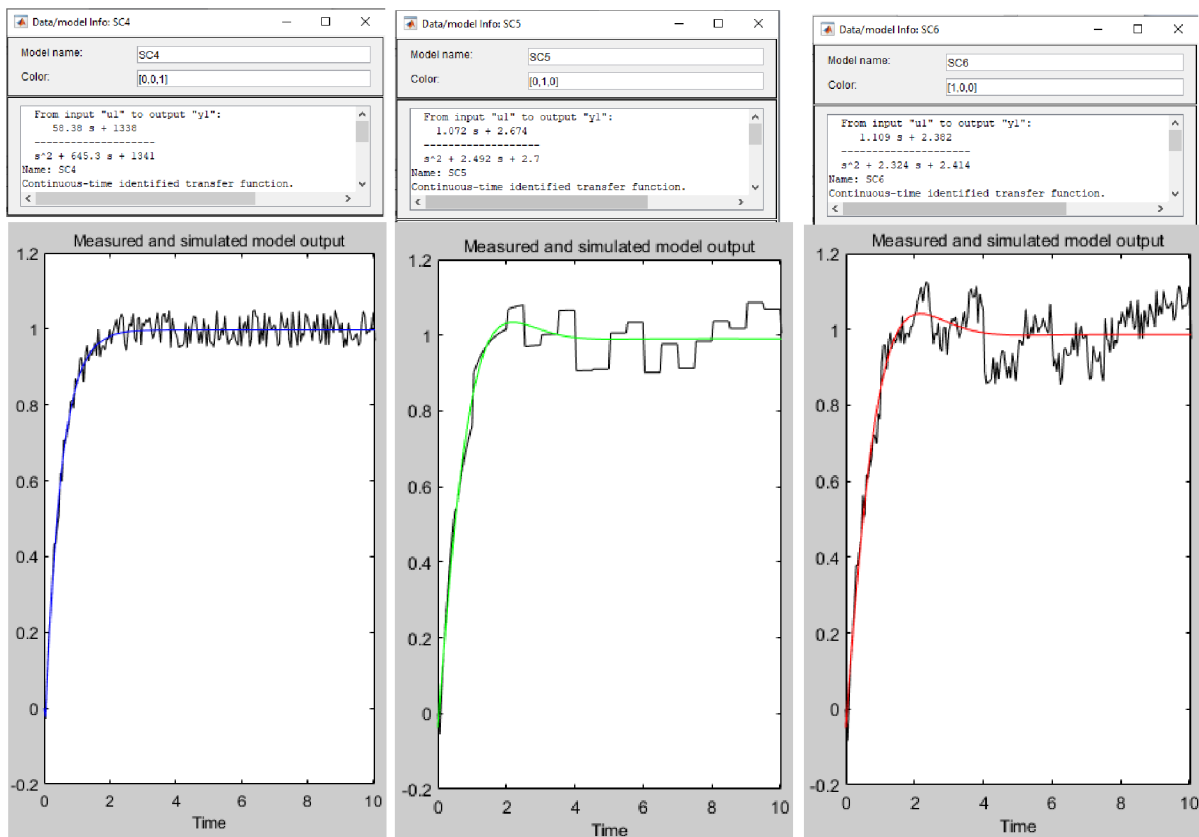
	MATLAB-System Identification Toolbox	Mobilní aplikace pro identifikaci a řízení signálů
SC1	0.7882	0.6596
SC2	2.3712	0.2617
SC3	3.1586	0.9406

Tab. 4.2 Suma odchylek od původního signálu A)

B) ODR druhého řádu levé strany a prvního řádu pravé strany:



Obr. 4.22 Výsledek identifikace verifikačních signálů B) mobilní aplikací (zleva: vysokofrekvenční šum, nízkofrekvenční šum, kombinace vysokofrekvenčního a nízkofrekvenčního šumu)



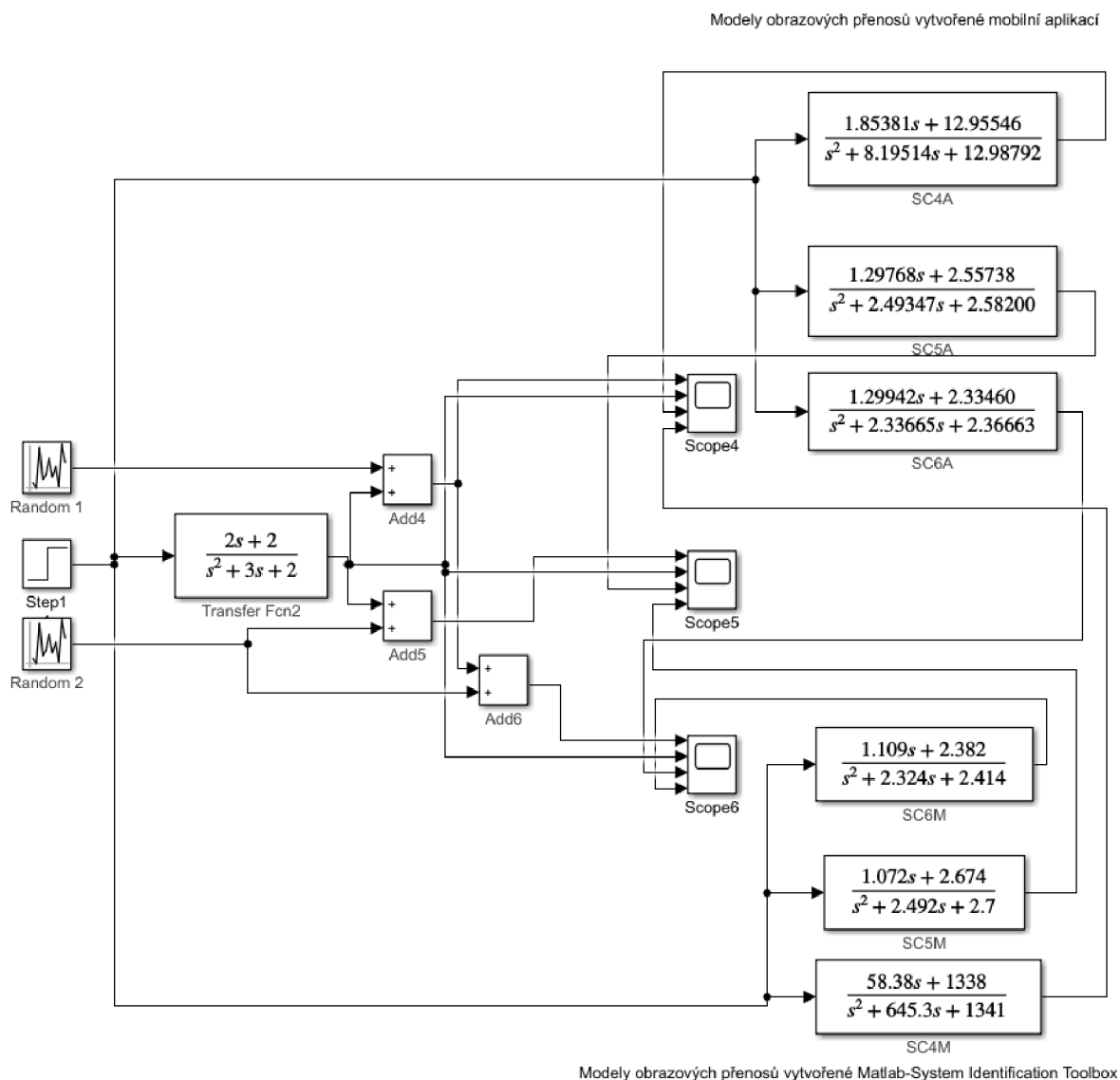
Obr. 4.23 Výsledek identifikace verifikačních signálů B) Matlab - Systém Identification Toolbox (zleva: vysokofrekvenční šum, nízkofrekvenční šum, kombinace vysokofrekvenčního a nízkofrekvenčního šumu)

Výsledky identifikace:

	MATLAB-System Identification Toolbox	Mobilní aplikace pro identifikaci a řízení signálů
SC4	$G(s) = \frac{58.38s + 1338}{s^2 + 645.3s + 1341}$	$G(s) = \frac{1.85381s + 12.95546}{s^2 + 8.19514s + 12.98792}$
SC5	$G(s) = \frac{1.072s + 2.674}{s^2 + 2.492s + 2.7}$	$G(s) = \frac{1.29768s + 2.55738}{s^2 + 2.49347s + 2.58200}$
SC6	$G(s) = \frac{1.109s + 2.382}{s^2 + 2.324s + 2.414}$	$G(s) = \frac{1.29942s + 2.33460}{s^2 + 2.33665s + 2.36663}$

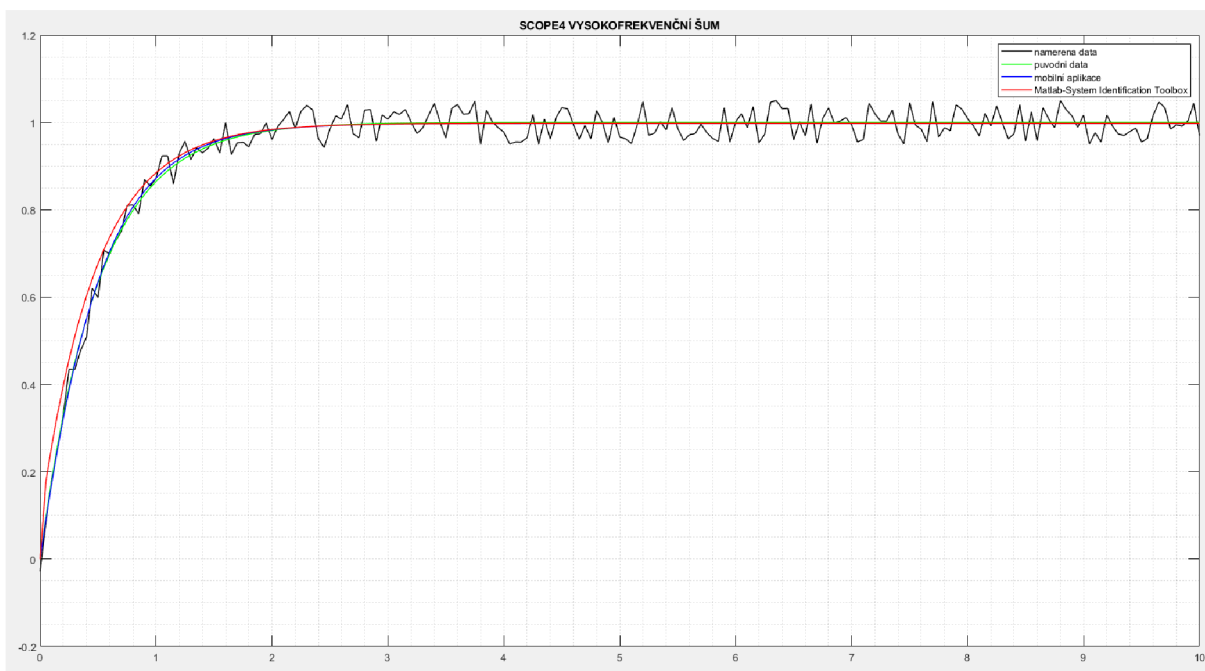
Tab. 4.3 Porovnání výsledků identifikace B)

Verifikační schéma Matlab Simulink:

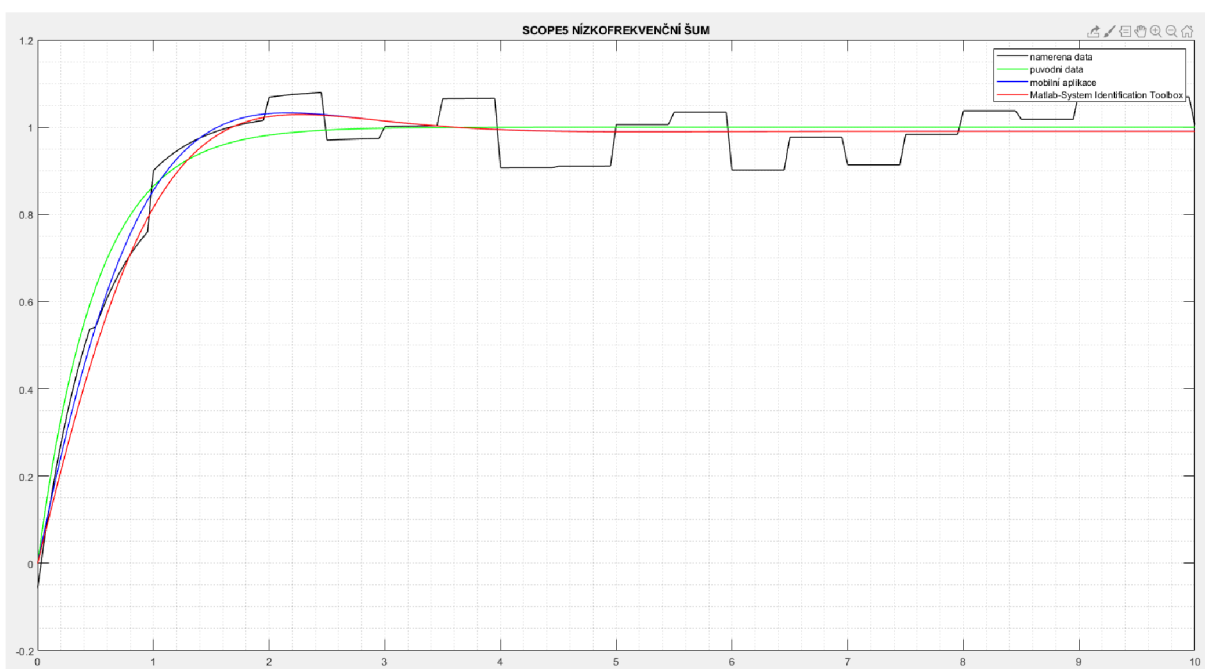


Obr. 4.24 Verifikační schéma B) identifikace/optimalizace Matlab Simulink

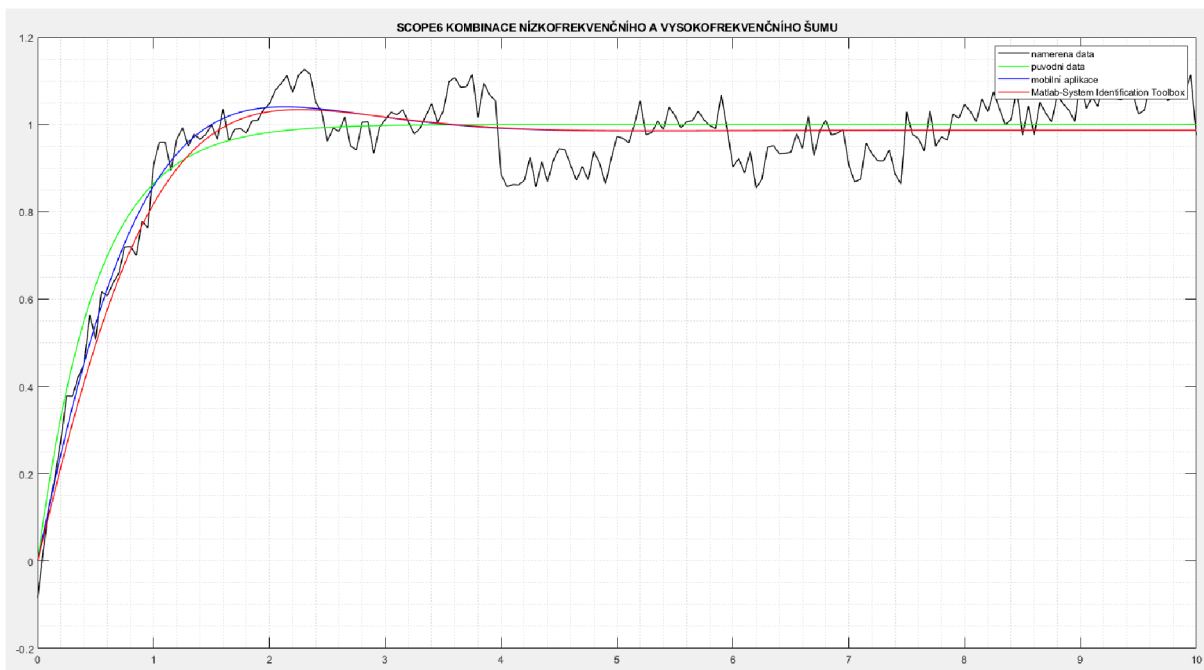
Grafické zobrazení výsledků verifikace identifikace:



Obr. 4.25 Grafické zobrazení výsledků verifikace identifikace B) vysokofrekvenční šum



Obr. 4.26 Grafické zobrazení výsledků verifikace identifikace B) nízkofrekvenční šum



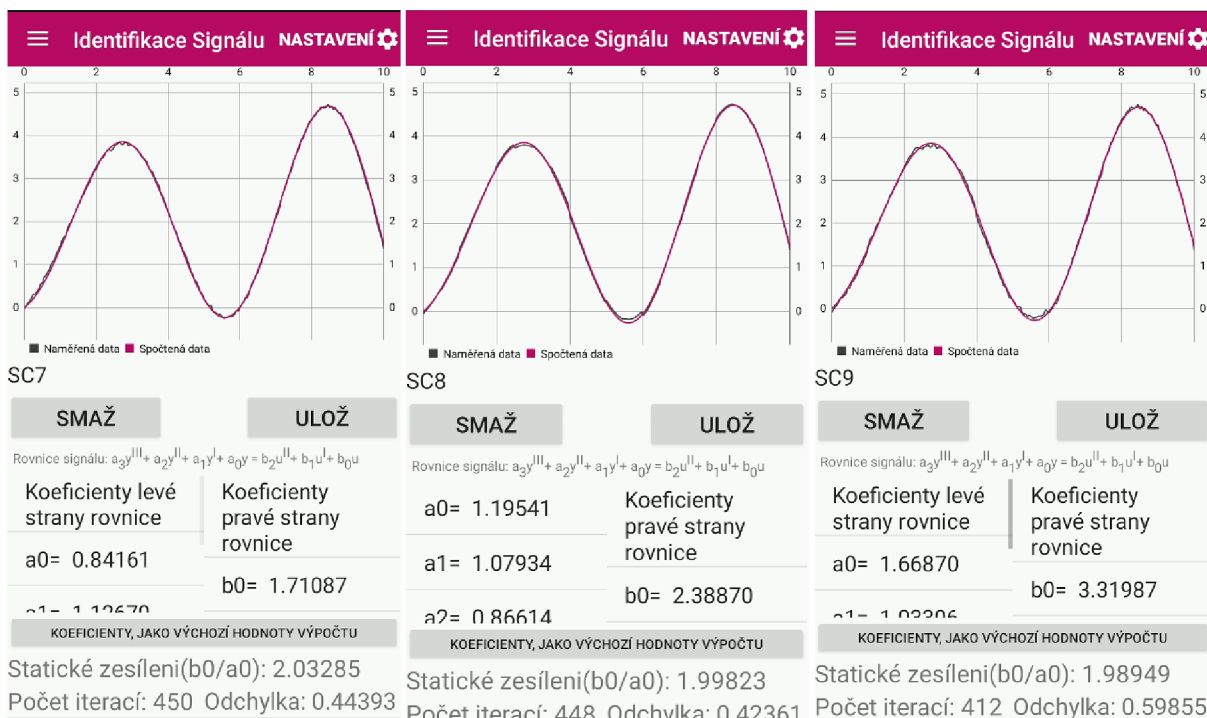
Obr. 4.27 Grafické zobrazení výsledků verifikace identifikace B) kombinace vysokofrekvenčního a nízkofrekvenčního šumu

Suma absolutních odchylek, od původního signálu:

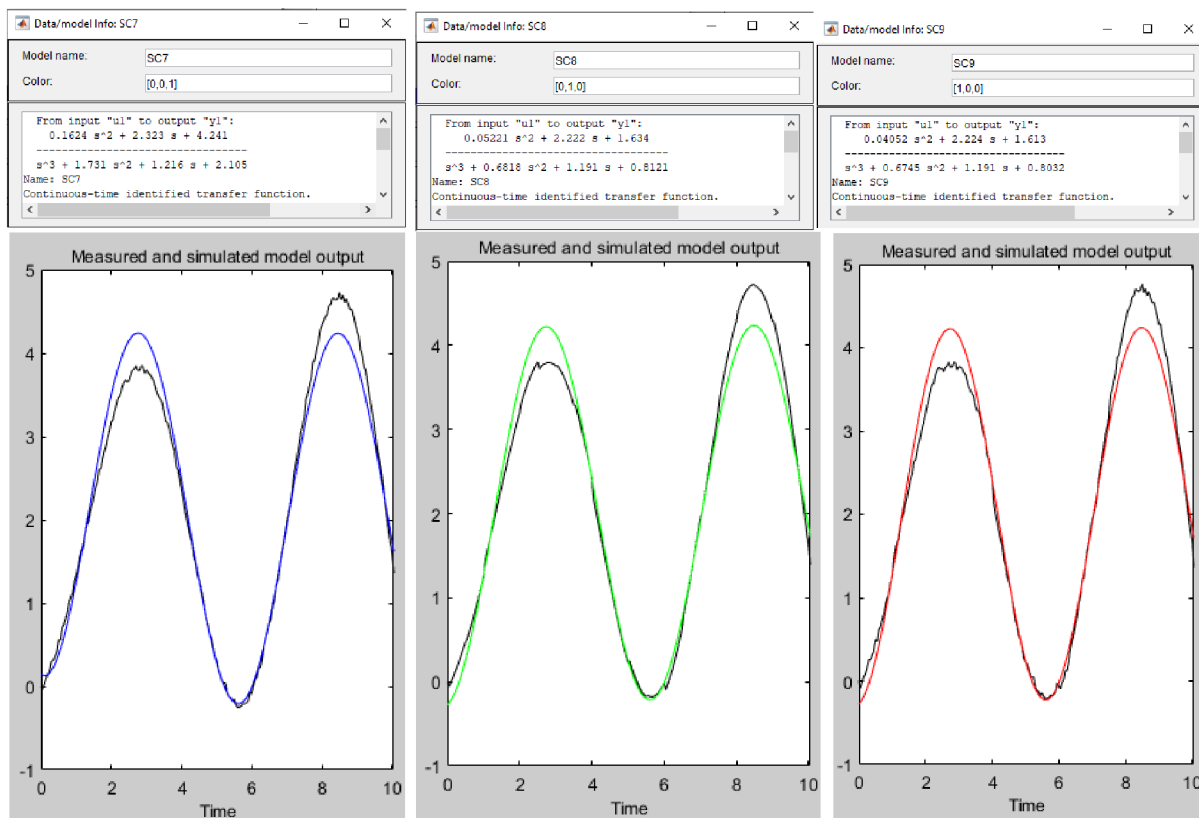
	MATLAB-System Identification Toolbox	Mobilní aplikace pro identifikaci a řízení signálů
SC4	1.4140	0.5635
SC5	4.5636	3.9417
SC6	5.0869	4.6755

Tab. 4.4 Suma odchylek od původního signálu B)

C) ODR třetího řádu levé strany a druhého řádu pravé strany:



Obr. 4.28 Výsledek identifikace verifikačních signálů C) mobilní aplikací (zleva: vysokofrekvenční šum, nízkofrekvenční šum, kombinace vysokofrekvenčního a nízkofrekvenčního šumu)



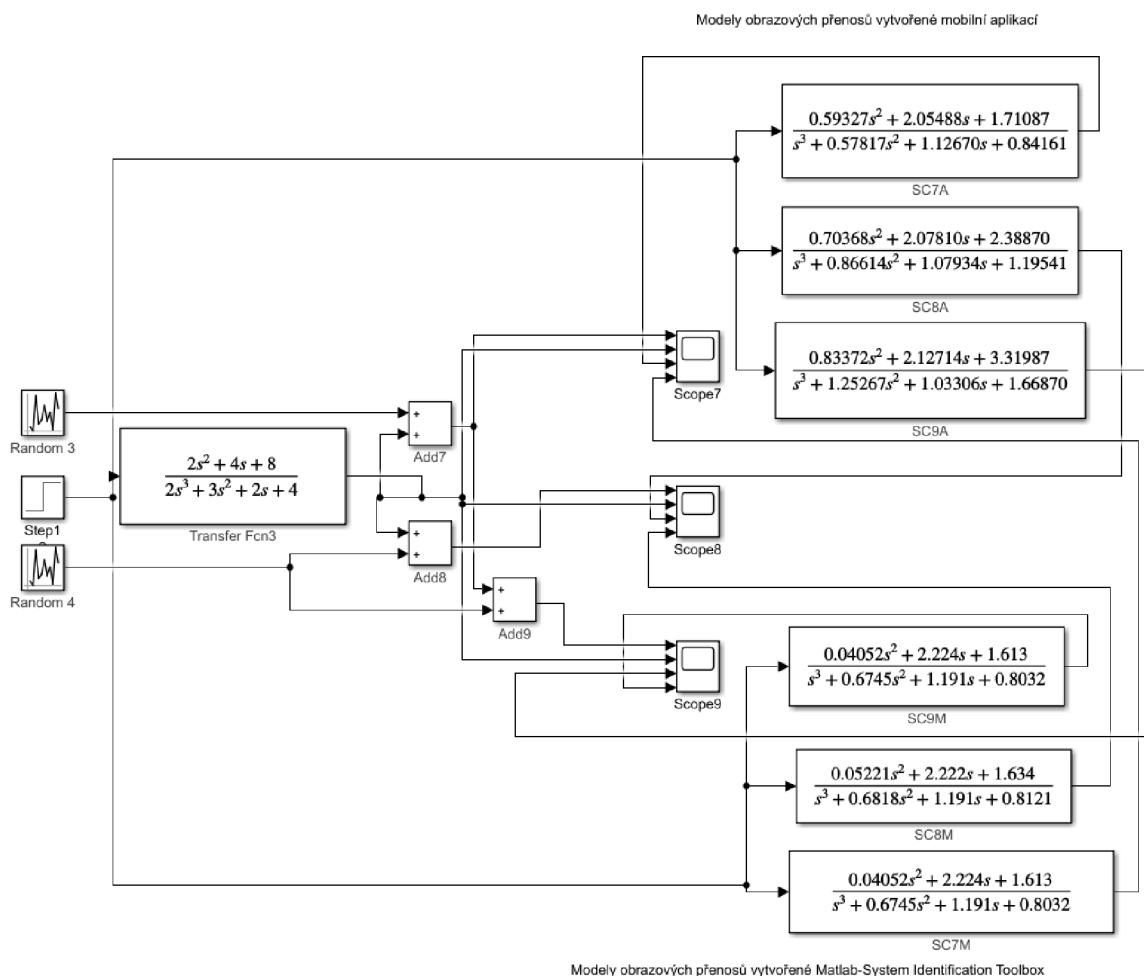
Obr. 4.29 Výsledek identifikace verifikačních signálů C) Matlab - System Identification Toolbox (zleva: vysokofrekvenční šum, nízkofrekvenční šum, kombinace vysokofrekvenčního a nízkofrekvenčního šumu)

Výsledky identifikace:

	MATLAB-System Identification Toolbox	Mobilní aplikace pro identifikaci a řízení signálů
SC7	$G(s) = \frac{0.1624s^2 + 2.323s + 4.241}{s^3 + 1.731s^2 + 1.216s + 2.105}$	$G(s) = \frac{0.59327s^2 + 2.05488s + 1.71087}{s^3 + 0.57817s^2 + 1.12670s + 0.84161}$
SC8	$G(s) = \frac{0.05221s^2 + 2.222s + 1.634}{s^3 + 0.6818s^2 + 1.191s + 0.8121}$	$G(s) = \frac{0.70368s^2 + 2.07810s + 2.38870}{s^3 + 0.86614s^2 + 1.07934s + 1.19541}$
SC9	$G(s) = \frac{0.04052s^2 + 2.224s + 1.613}{s^3 + 0.6745s^2 + 1.191s + 0.8032}$	$G(s) = \frac{0.83372s^2 + 2.12714s + 3.31987}{s^3 + 1.25267s^2 + 1.03306s + 1.66870}$

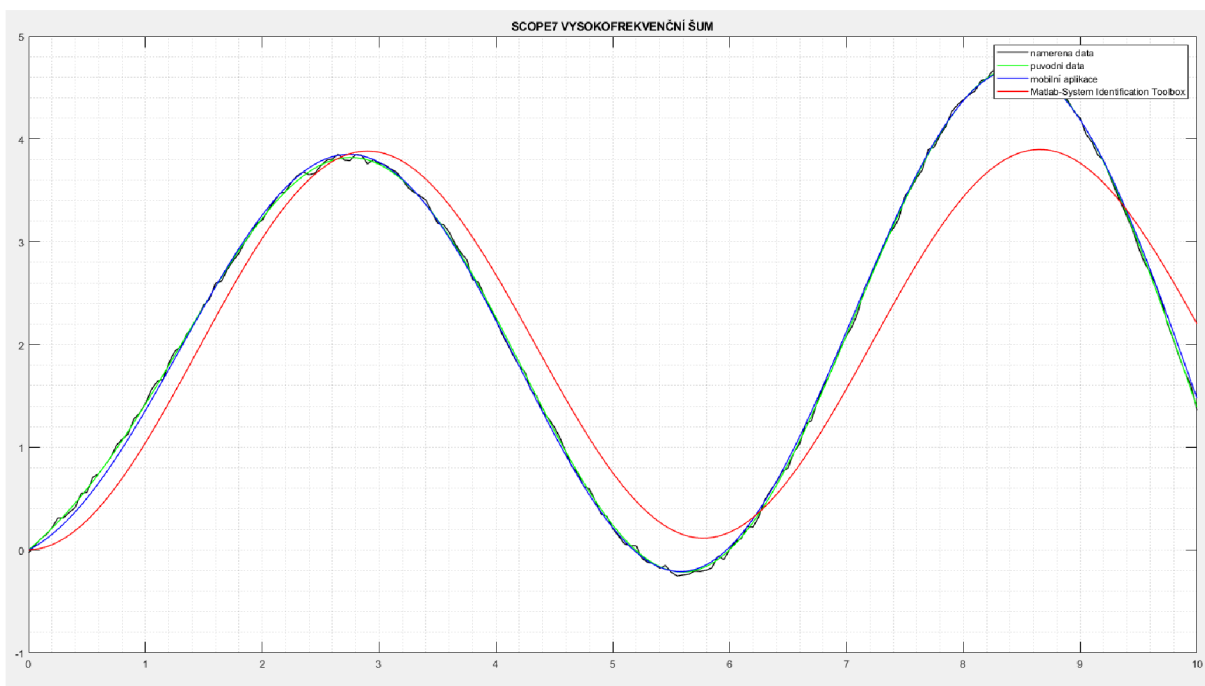
Tab. 4.5 Porovnání výsledků identifikace C)

Verifikační schéma Matlab Simulink:

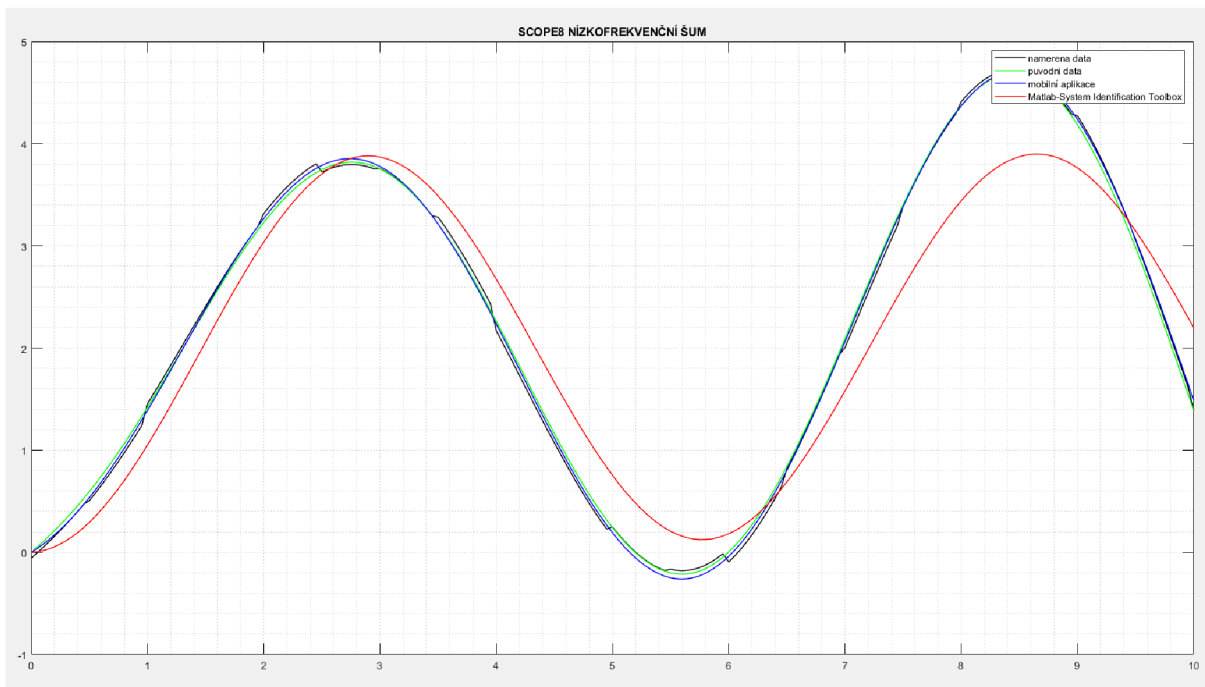


Obr. 4.30 Verifikační schéma C) identifikace/optimalizace Matlab Simulink

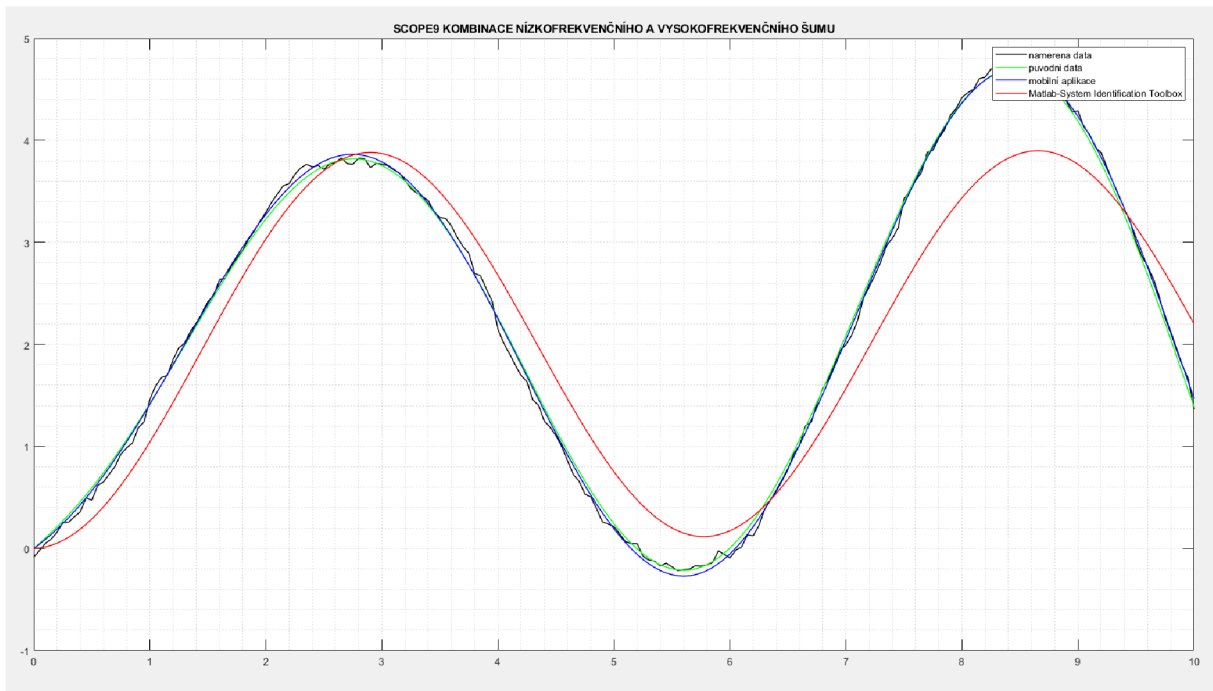
Grafické zobrazení výsledků verifikace identifikace:



Obr. 4.31 Grafické zobrazení výsledků verifikace identifikace C) vysokofrekvenční šum



Obr. 4.32 Grafické zobrazení výsledků verifikace identifikace C) nízkofrekvenční šum



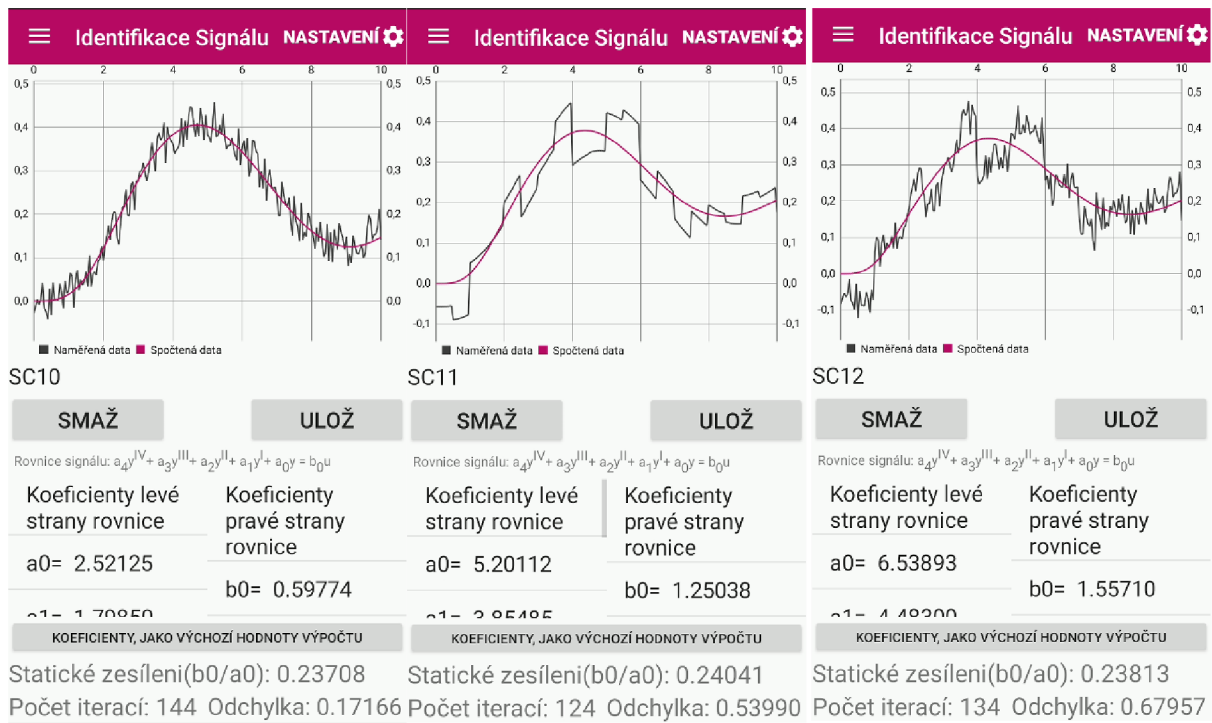
Obr. 4.33 Grafické zobrazení výsledků verifikace identifikace C) kombinace vysokofrekvenčního a nízkofrekvenčního šumu

Suma absolutních odchylek, od původního signálu:

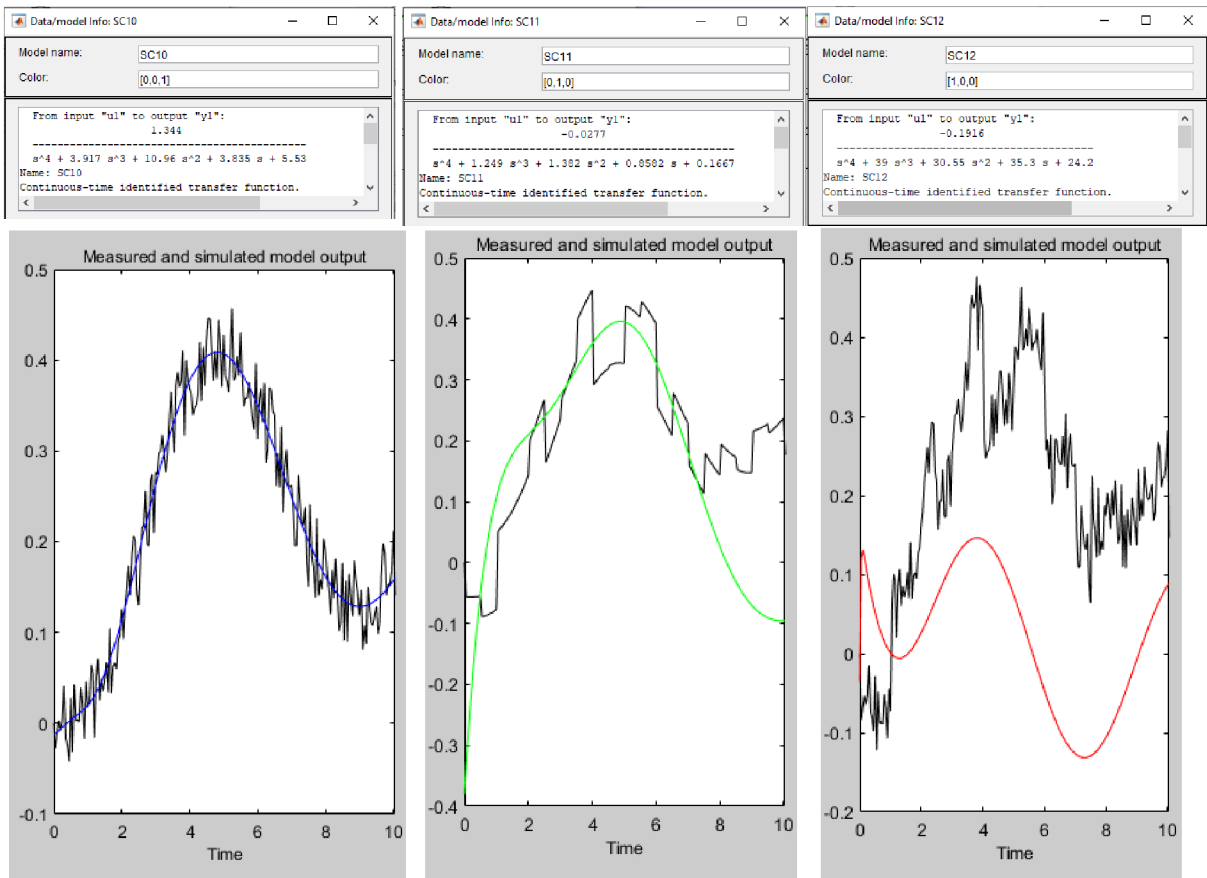
	MATLAB-System Identification Toolbox	Mobilní aplikace pro identifikaci a řízení signálů
SC7	78.5024	5.9166
SC8	78.0447	7.3182
SC9	78.5024	7.3176

Tab. 4.6 Suma odchylek od původního signálu C)

D) ODR čtvrtého řádu levé strany a nultého řádu pravé strany:



Obr. 4.34 Výsledek identifikace verifikačních signálů D) mobilní aplikací (zleva: vysokofrekvenční šum, nízkofrekvenční šum, kombinace vysokofrekvenčního a nízkofrekvenčního šumu)



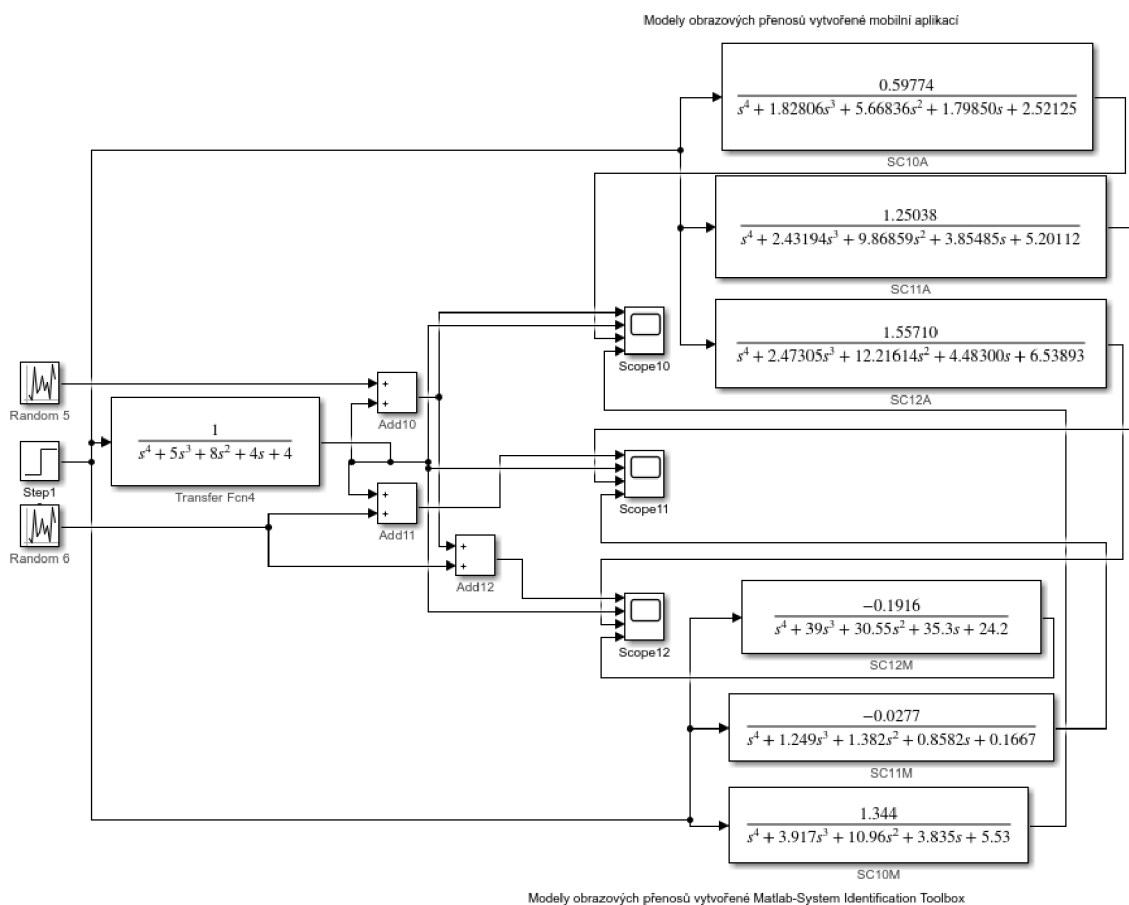
Obr. 4.35 Výsledek identifikace verifikačních signálů D) Matlab - System Identification Toolbox (zleva: vysokofrekvenční šum, nízkofrekvenční šum, kombinace vysokofrekvenčního a nízkofrekvenčního šumu)

Výsledky identifikace:

	MATLAB-System Identification Toolbox	Mobilní aplikace pro identifikaci a řízení signálů
SC10	$G(s) = \frac{1.34}{s^4 + 3.92s^3 + 10.96s^2 + 3.84s + 5.53}$	$G(s) = \frac{0.60}{s^4 + 1.83s^3 + 5.67s^2 + 1.79s + 2.52}$
SC11	$G(s) = \frac{-0.03}{s^4 + 1.25s^3 + 1.38s^2 + 0.86s + 0.17}$	$G(s) = \frac{1.25}{s^4 + 2.43s^3 + 9.87s^2 + 3.85s + 5.20}$
SC12	$G(s) = \frac{-0.19}{s^4 + 39s^3 + 30.55s^2 + 35.3s + 24.2}$	$G(s) = \frac{1.56}{s^4 + 2.47s^3 + 12.22s^2 + 4.48s + 6.54}$

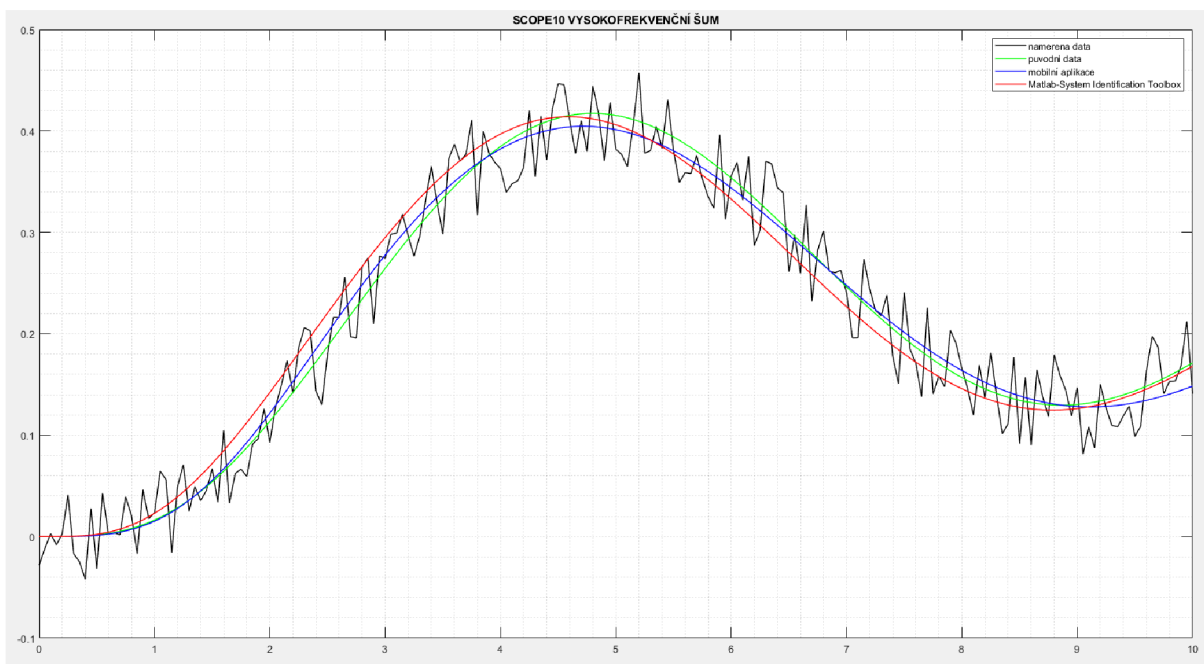
Tab. 4.7 Porovnání výsledků identifikace D)

Verifikační schéma Matlab Simulink:

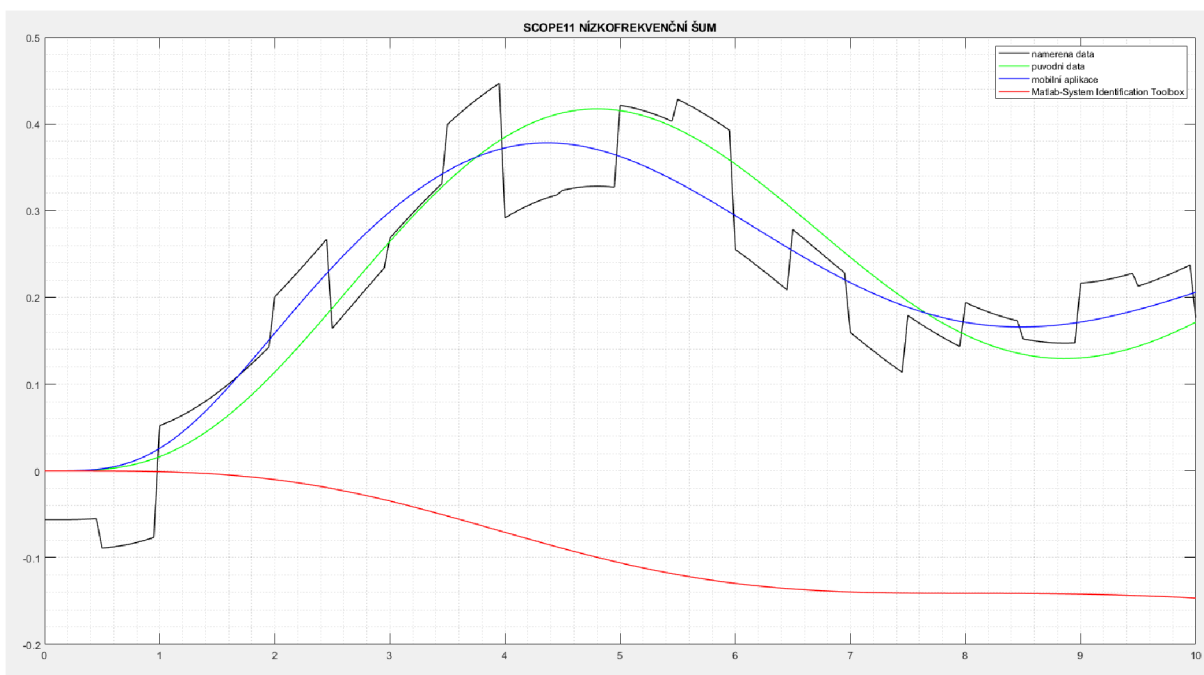


Obr. 4.36 Verifikační schéma D) identifikace/optimalizace Matlab Simulink

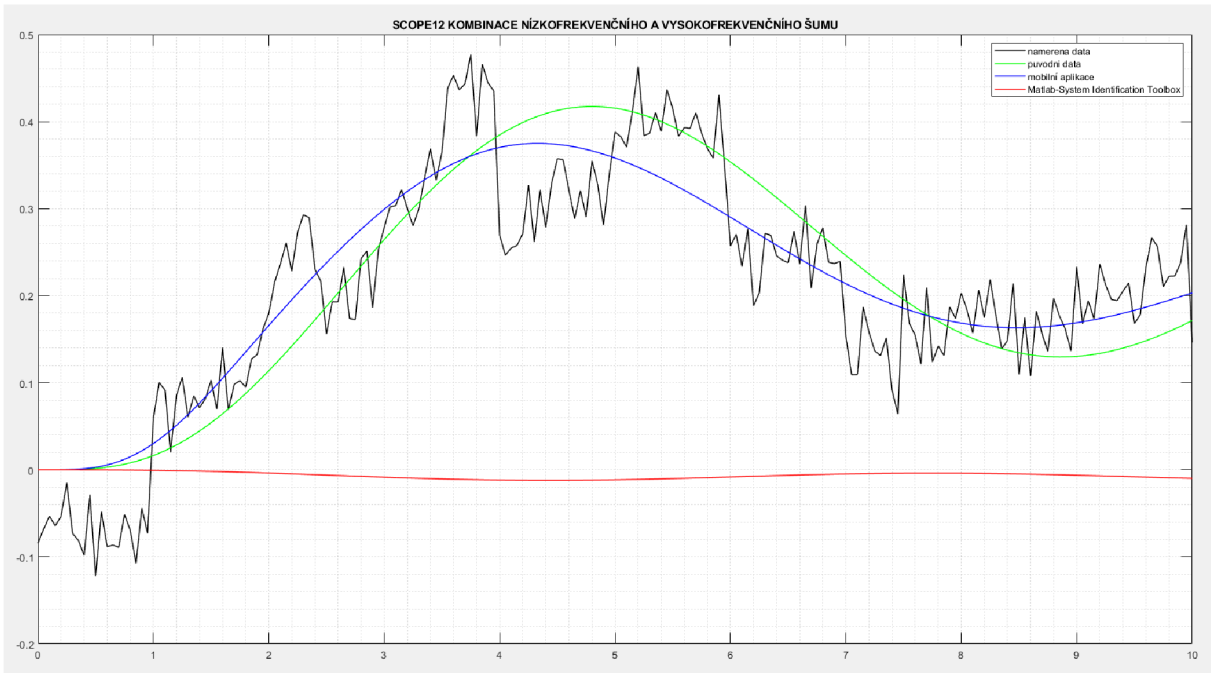
Grafické zobrazení výsledků verifikace identifikace:



Obr. 4.37 Grafické zobrazení výsledků verifikace identifikace D) vysokofrekvenční šum



Obr. 4.38 Grafické zobrazení výsledků verifikace identifikace D) nízkofrekvenční šum



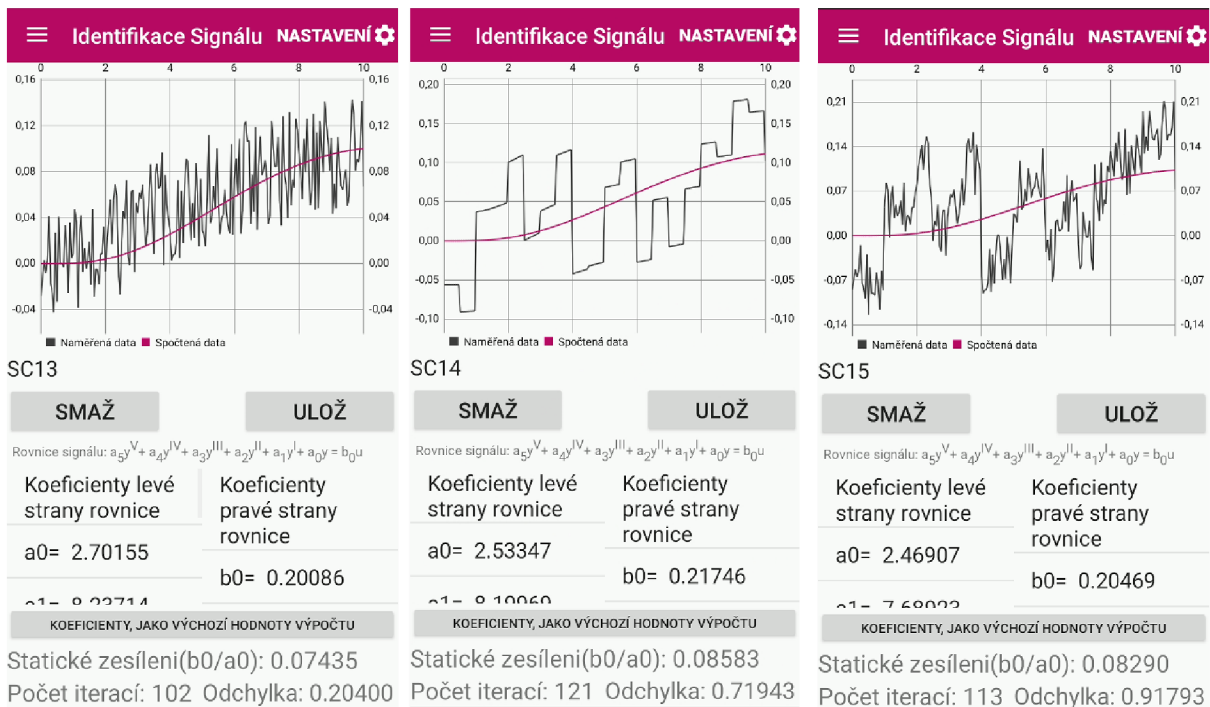
Obr. 4.39 Grafické zobrazení výsledků verifikace identifikace D) kombinace vysokofrekvenčního a nízkofrekvenčního šumu

Suma absolutních odchylek, od původního signálu:

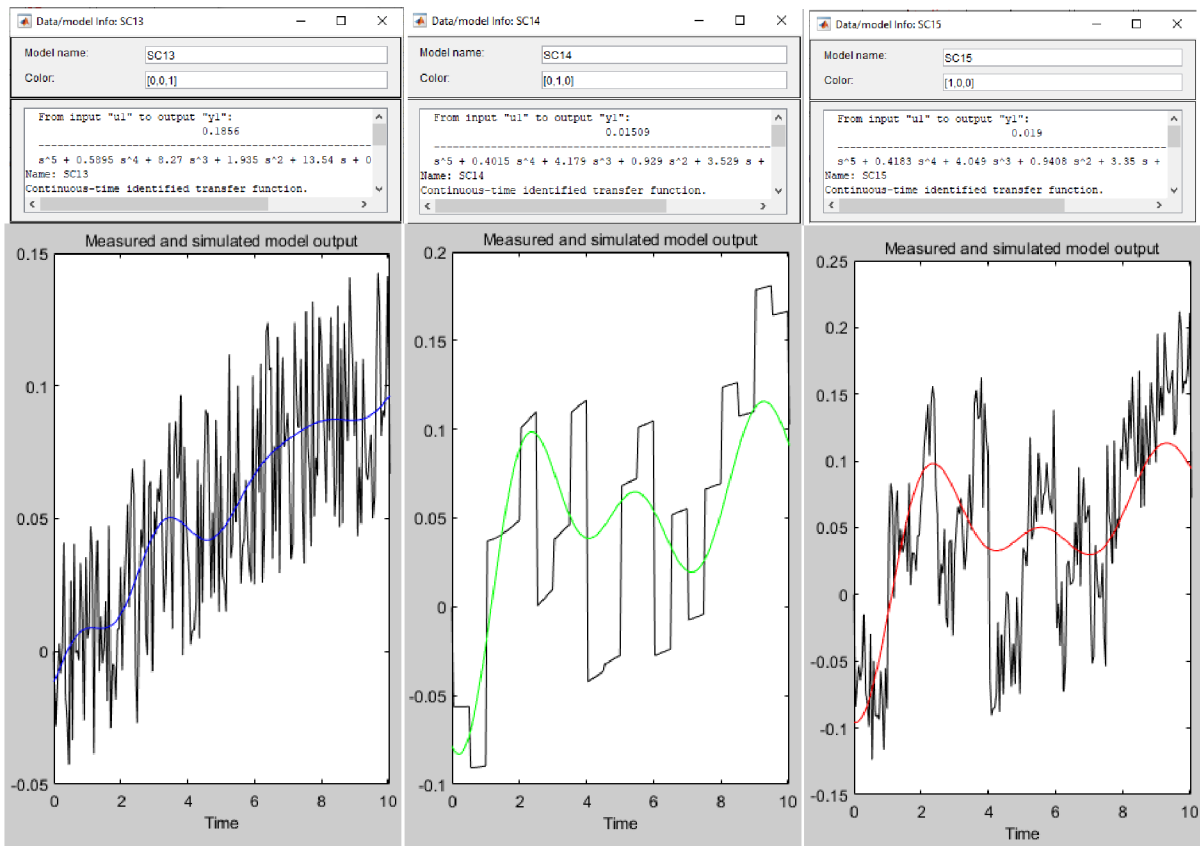
	MATLAB-System Identification Toolbox	Mobilní aplikace pro identifikaci a řízení signálů
SC10	2.8437	1.4198
SC11	/	6.2943
SC12	/	6.6753

Tab. 4.8 Suma odchylek od původního signálu D)

E) ODR pátého řádu levé strany a nultého řádu pravé strany:



Obr. 4.40 Výsledek identifikace verifikačních signálů E) mobilní aplikací (zleva: vysokofrekvenční šum, nízkofrekvenční šum, kombinace vysokofrekvenčního a nízkofrekvenčního šumu)

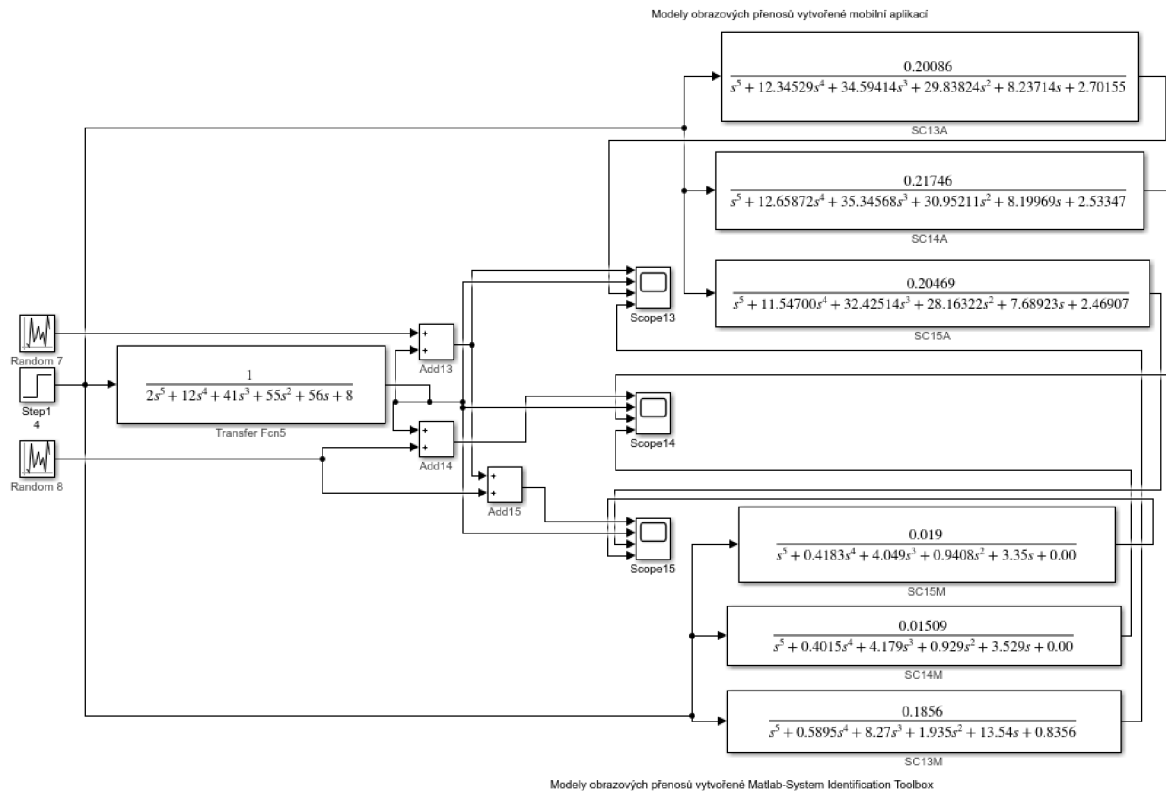


Obr. 4.41 Výsledek identifikace verifikačních signálů E) Matlab - System Identification Toolbox (zleva: vysokofrekvenční šum, nízkofrekvenční šum, kombinace vysokofrekvenčního a nízkofrekvenčního šumu)

Výsledky identifikace:

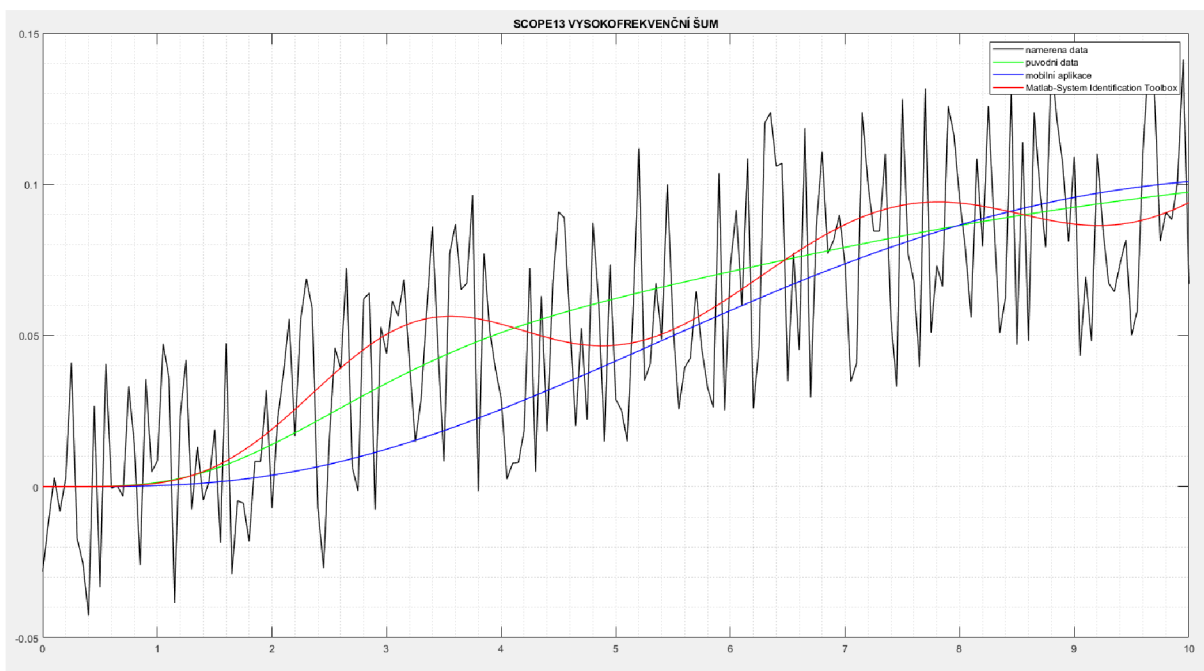
	MATLAB-System Identification Toolbox	Mobilní aplikace pro identifikaci a řízení signálů
SC13	$G(s) = \frac{0.19}{s^5 + 0.59s^4 + 8.27s^3 + 1.94s^2 + 13.54s + 0.84}$	$G(s) = \frac{0.20}{s^5 + 12.35s^4 + 34.59s^3 + 29.84s^2 + 8.24s + 2.70}$
SC14	$G(s) = \frac{0.02}{s^5 + 0.40s^4 + 4.18s^3 + 0.93s^2 + 3.53s + 0.00}$	$G(s) = \frac{0.22}{s^5 + 12.66s^4 + 35.35s^3 + 30.95s^2 + 8.20s + 2.53}$
SC15	$G(s) = \frac{0.02}{s^5 + 0.42s^4 + 4.05s^3 + 0.94s^2 + 3.35s + 0.00}$	$G(s) = \frac{0.20}{s^5 + 11.55s^4 + 32.43s^3 + 28.16s^2 + 7.69s + 2.47}$

Tab. 4.9 Porovnání výsledků identifikace E)

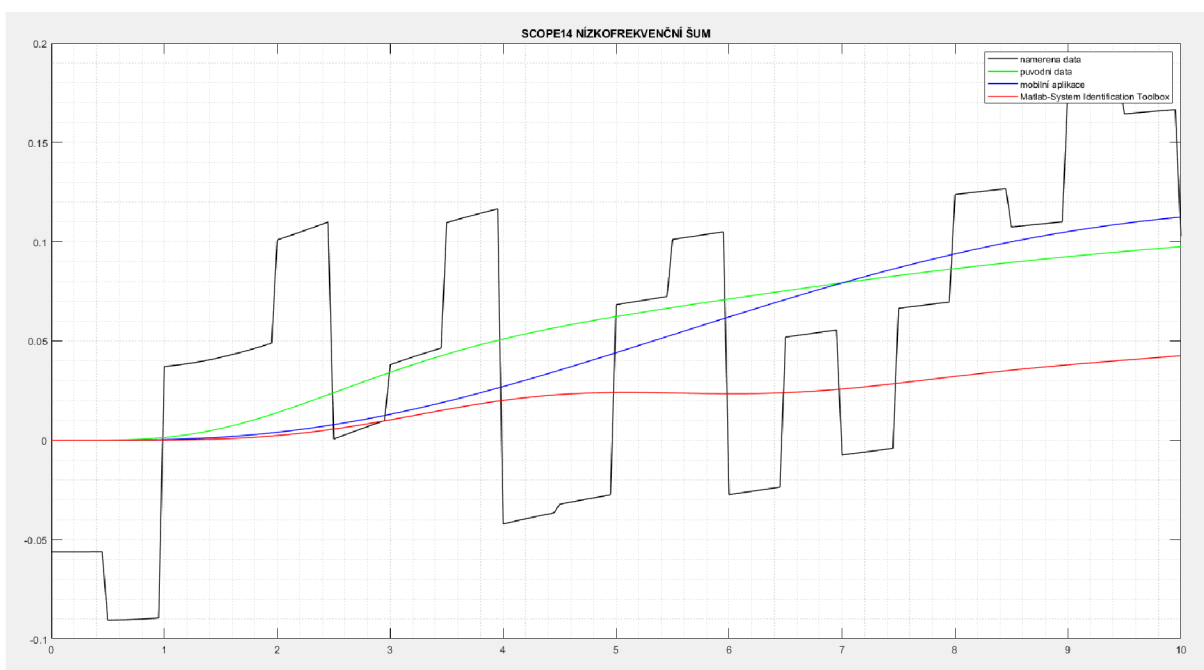


Obr. 4.42 Verifikační schéma E) identifikace/optimalizace Matlab Simulink

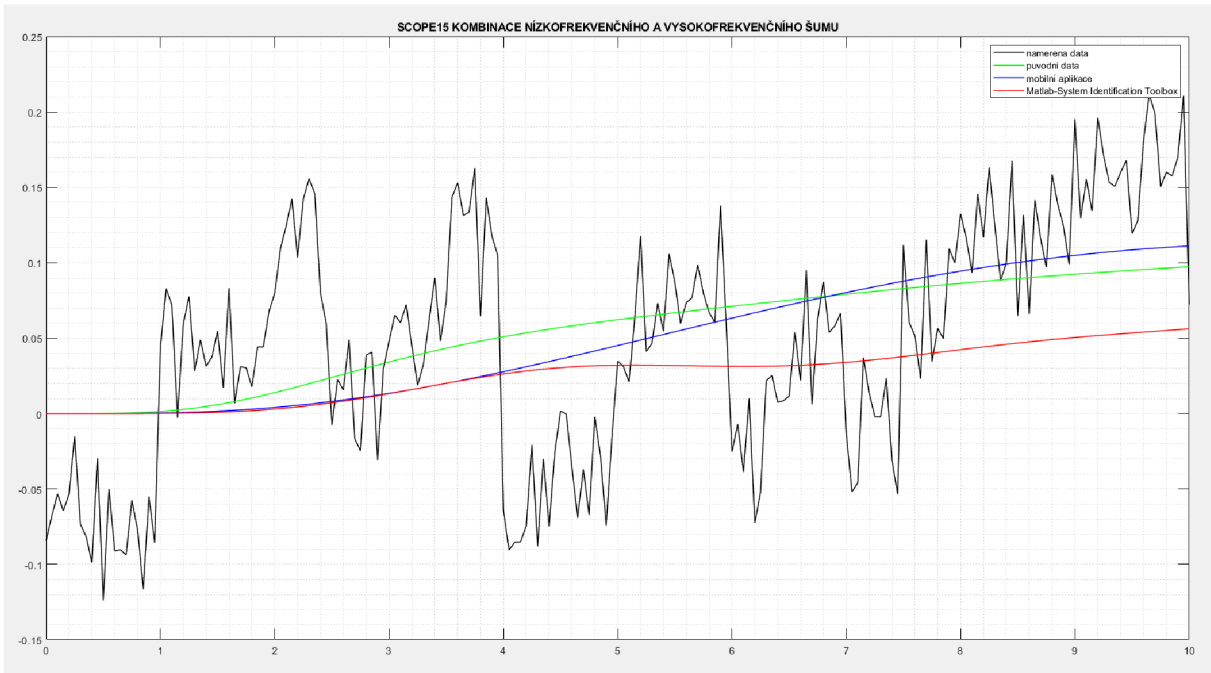
Grafické zobrazení výsledků verifikace identifikace:



Obr. 4.43 Grafické zobrazení výsledků verifikace identifikace E) vysokofrekvenční šum



Obr. 4.44 Grafické zobrazení výsledků verifikace identifikace E) nízkofrekvenční šum



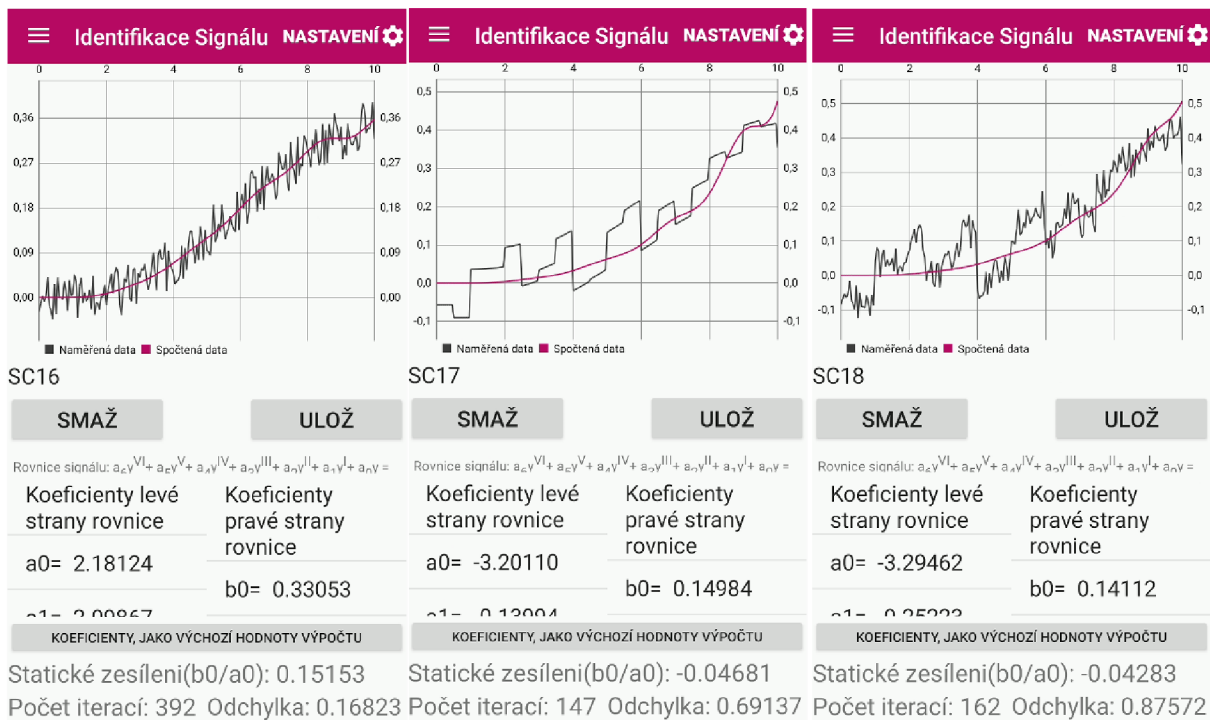
Obr. 4.45 Grafické zobrazení výsledků verifikace identifikace E) kombinace vysokofrekvenčního a nízkofrekvenčního šumu

Suma absolutních odchylek, od původního signálu:

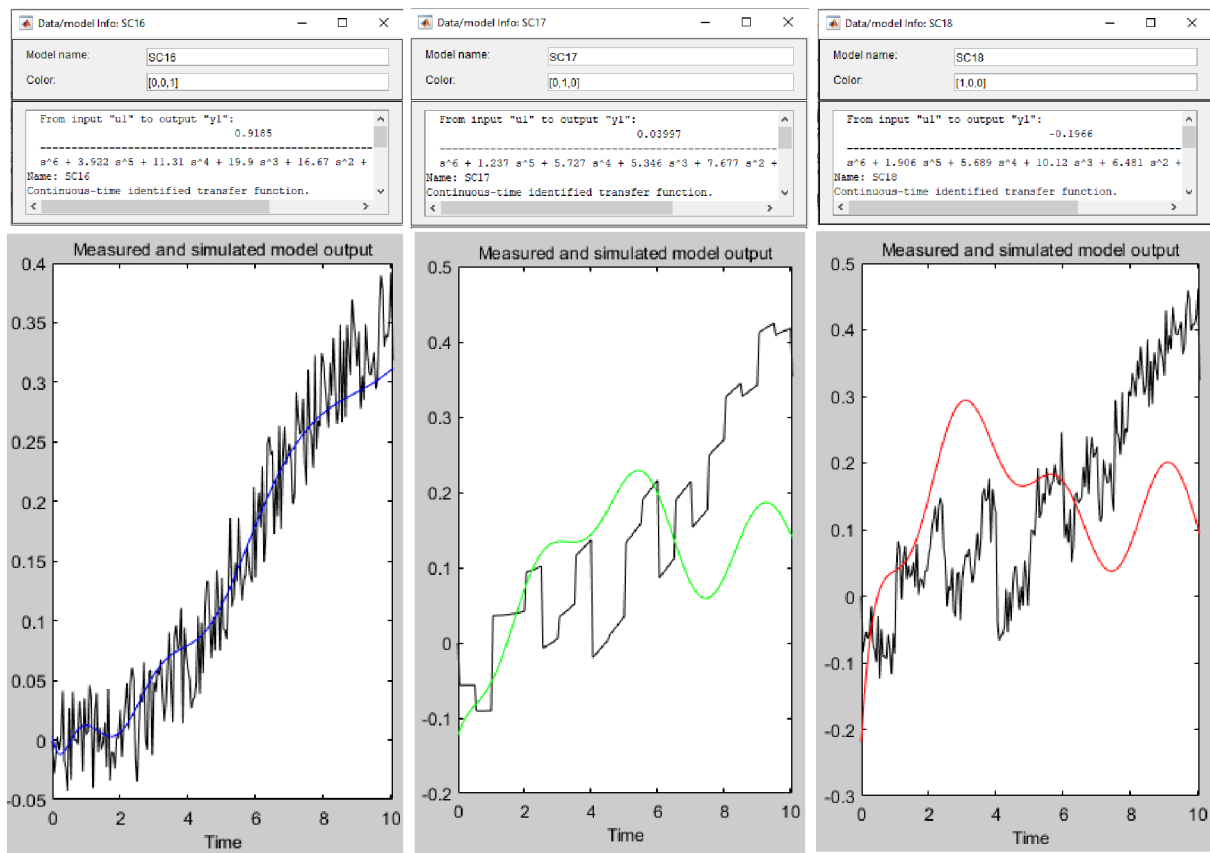
	MATLAB-System Identification Toolbox	Mobilní aplikace pro identifikaci a řízení signálů
SC13	1.4085	2.0664
SC14	6.8886	2.2469
SC15	5.6136	2.1797

Tab. 4.10 Suma odchylek od původního signálu E)

F) ODR šestého řádu levé strany a nultého řádu pravé strany:



Obr. 4.46 Výsledek identifikace verifikačních signálů F) mobilní aplikací (zleva: vysokofrekvenční šum, nízkofrekvenční šum, kombinace vysokofrekvenčního a nízkofrekvenčního šumu)



Obr. 4.47 Výsledek identifikace verifikačních signálů F) Matlab - System Identification Toolbox (zleva: vysokofrekvenční šum, nízkofrekvenční šum, kombinace vysokofrekvenčního a nízkofrekvenčního šumu)

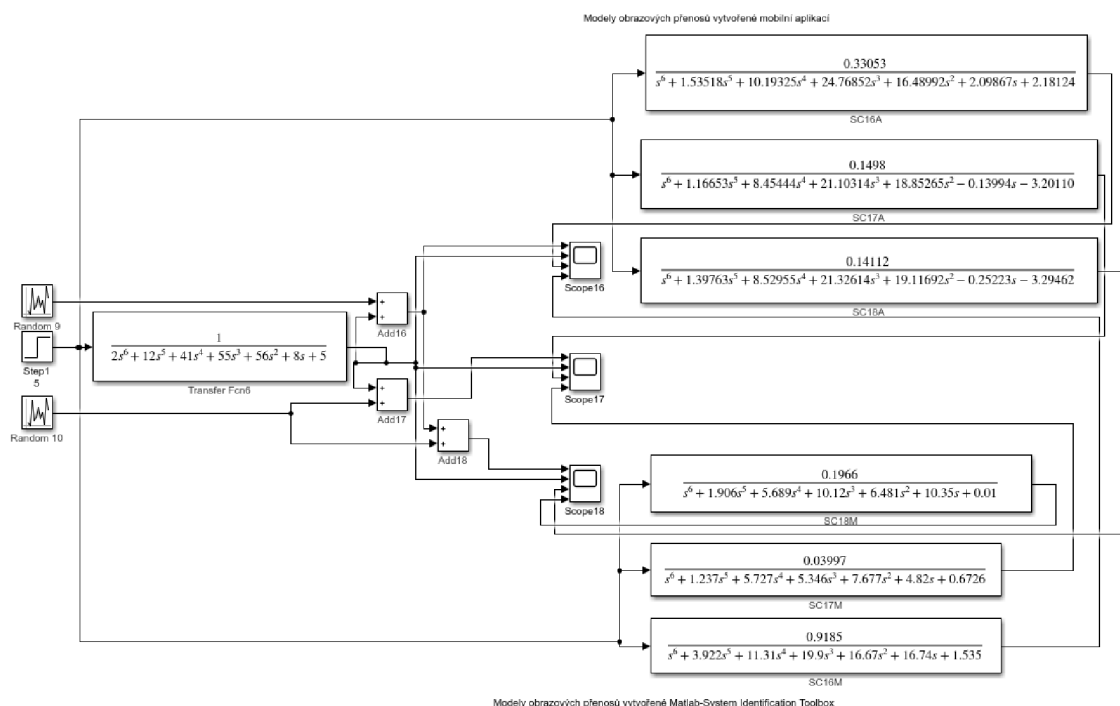
Výsledky identifikace:

	MATLAB-System Identification Toolbox	Mobilní aplikace pro identifikaci a řízení signálů
SC16	$G(s) = \frac{0.92}{s^6 + 3.92s^5 + 11.31s^4 + 19.9s^3 + 16.67s^2 + 16.74s + 1.54}$	$G(s) = \frac{0.33}{s^6 + 1.54s^5 + 10.19s^4 + 24.77s^3 + 16.49s^2 + 2.10s + 2.18}$
SC17	$G(s) = \frac{0.04}{s^6 + 1.24s^5 + 5.727s^4 + 5.35s^3 + 7.68s^2 + 4.82s + 0.67}$	$G(s) = \frac{0.15}{s^6 + 1.17s^5 + 8.45s^4 + 21.10s^3 + 18.85s^2 - 0.14s - 3.20}$
SC18	$G(s) = \frac{-0.20}{s^6 + 1.91s^5 + 5.69s^4 + 10.12s^3 + 6.48s^2 + 10.35s + 0.01}$	$G(s) = \frac{0.14}{s^6 + 1.40s^5 + 8.53s^4 + 21.33s^3 + 19.12s^2 - 0.25s - 3.29}$

Tab. 4.11 Porovnání výsledků identifikace F)

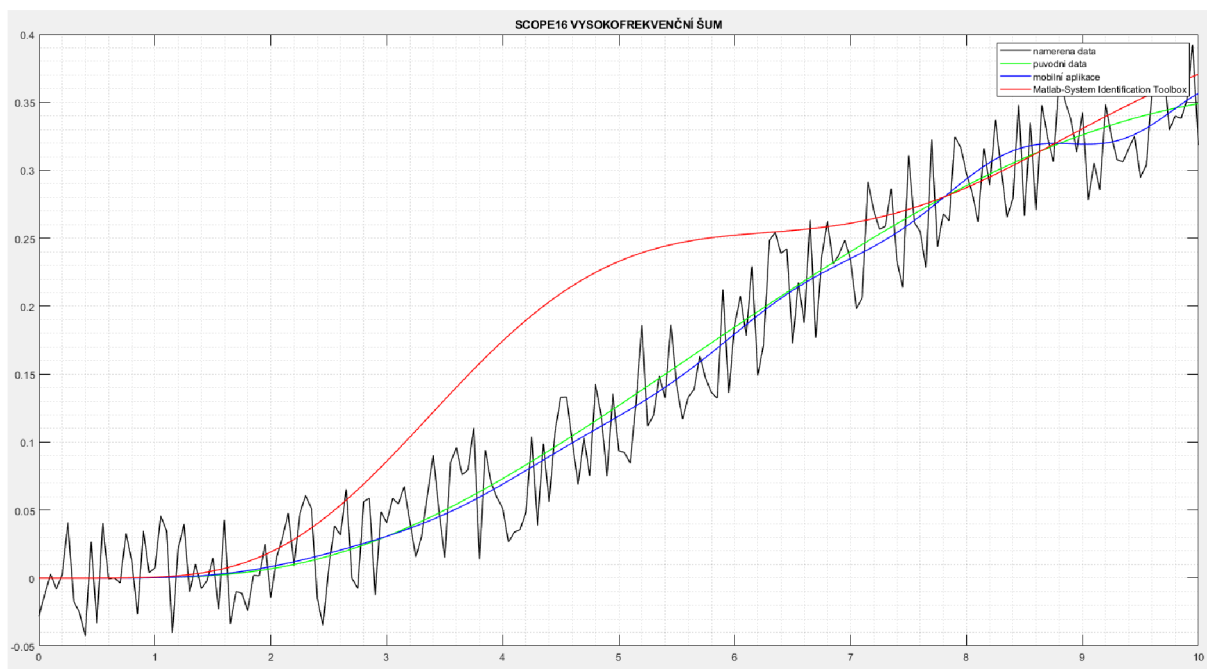
Pozn. přenosy SC17 a SC18 nelze použít k řízení systému PID regulátorem, kvůli střídavým znaménkům, ve jmenovateli přenosu, jedná se tedy o neplatný výsledek. Daný jev je způsoben velkým rozkmitem hodnot, množstvím koeficientů pro optimalizaci a neustálením dynamiky daného testovaného systému.

Verifikační schéma Matlab Simulink:

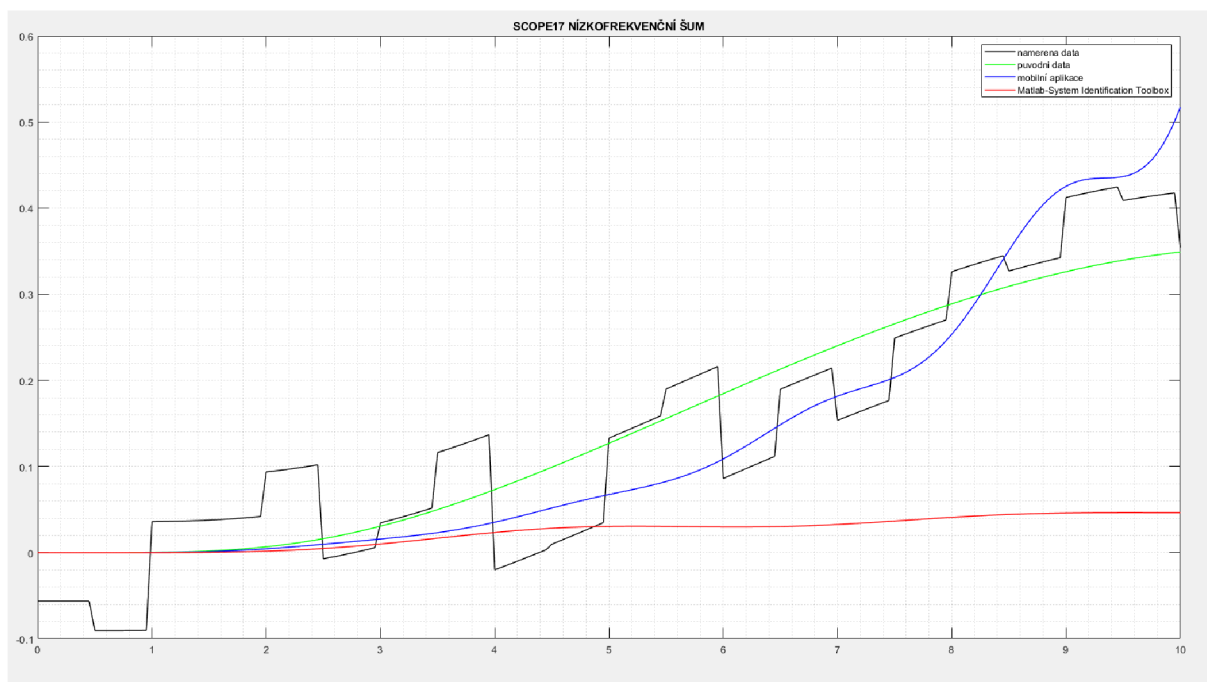


Obr. 4.48 Verifikační schéma F) identifikace/optimalizace Matlab Simulink

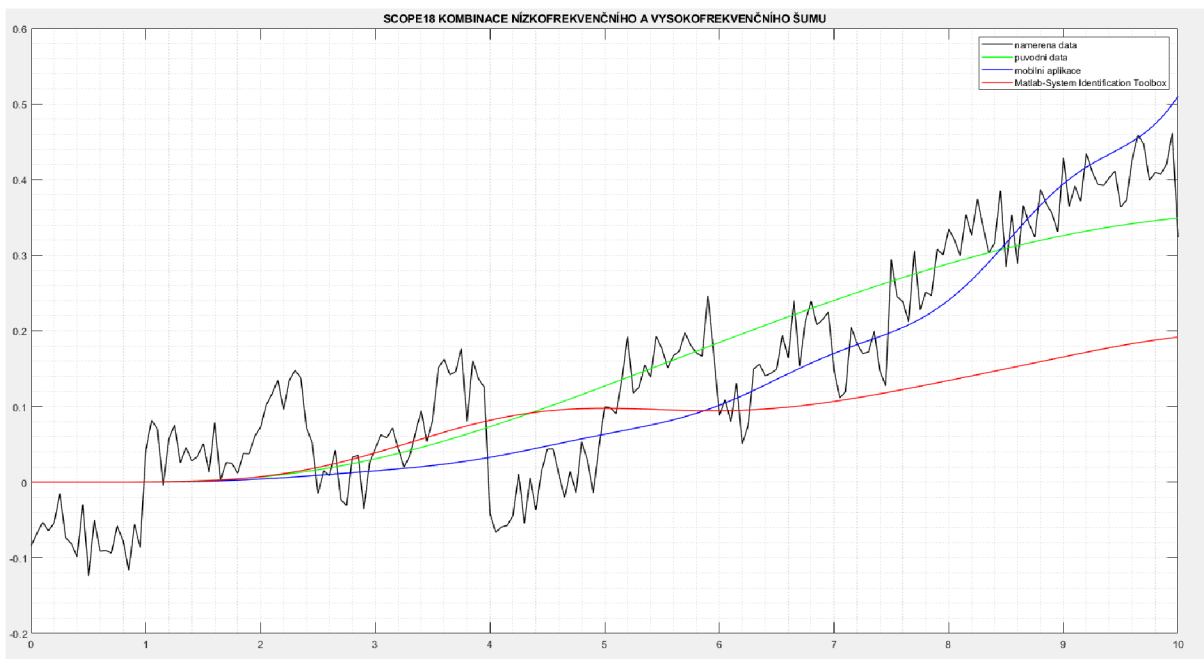
Grafické zobrazení výsledků verifikace identifikace:



Obr. 4.49 Grafické zobrazení výsledků verifikace identifikace F) vysokofrekvenční šum



Obr. 4.50 Grafické zobrazení výsledků verifikace identifikace F) nízkofrekvenční šum



Obr. 4.51 Grafické zobrazení výsledků verifikace identifikace F) kombinace vysokofrekvenčního a nízkofrekvenčního šumu

Suma absolutních odchylek, od původního signálu:

	MATLAB-System Identification Toolbox	Mobilní aplikace pro identifikaci a řízení signálů
SC16	7.5836	0.8046
SC17	24.3946	8.7489
SC18	13.3058	8.9537

Tab. 4.12 Suma odchylek od původního signálu F)

4.2.3 Shrnutí verifikace identifikace:

Jak je vidět, na výsledcích, a grafech výše. Aplikace si dokáže relativně dobře poradit s vysokofrekvenčním i nízkofrekvenčním šumem, popřípadě jejich kombinací. A výsledky identifikace jsou srovnatelné, v některých případech dokonce lepší než v případě použití Matlab System Identification Toolbox. Jak bylo demonstrováno, aplikace začíná mít problém s identifikací systémů šestého řádu a vyšších, záleží však opět na konkrétním systému a kvalitě naměřených dat. V praxi se však lze jen velmi zřídka setkat s nutností identifikovat systém modelem takto vysokého řádu.

4.3 VERIFIKACE SEŘIZOVÁNÍ PID REGULÁTORU

Pro verifikaci seřízení PID regulátoru, je opět využito modelů Matlab Simulink, ve kterém jsou vymodelovány základní typy PID regulátoru. A to úpravou simulačních schémat popsaných v obr. 4.1 přidáním bloků: Gain, Integrator, derivator a sum viz. níže.

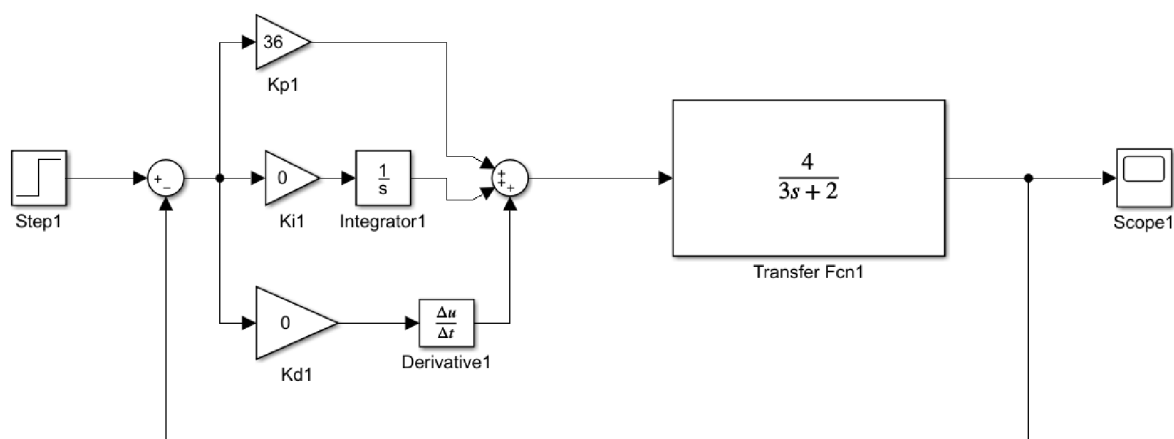
Takto vytvořeným modelem je zpracovávána příslušná přechodová funkce. Spočtený signál regulace je poté porovnáván s daty spočtenými mobilní aplikací. Takto získaná data byla pro porovnání přenesena do Matlabu, za použití vývojového prostředí Android Studio a jeho nástroje Logcat. Pro potřeby seřízení PID regulátoru mobilní aplikaci, byly do aplikace přímo zadány koeficienty příslušné seřizované ODR. Aby bylo seřízení co nejobektivnější.

Je třeba mít na paměti, že přesnost výpočtového mechanismu mobilní aplikace je pro simulaci spojitého PID regulátoru závislá, na velikosti použitého integračního kroku viz. 3.3. Proto je porovnání s mobilní aplikací provedeno vždy pro tři velikosti integračního kroku: 0.01, 0.001 a 0.0001 viz. níže. Přičemž data z vygenerována Matlab Simulink jsou brána jako referenční.

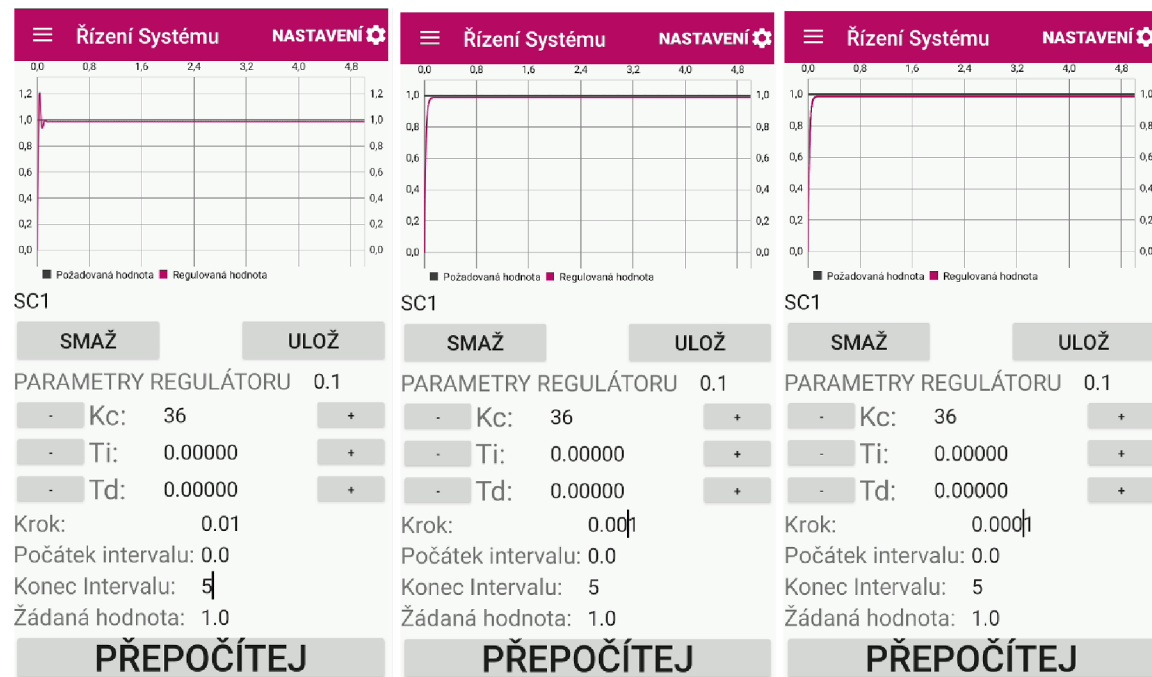
Za účelem zmenšení rozsahu práce, bylo kompletní porovnání seřízení všech typů regulátorů provedeno pouze pro první ODR. Zbylé ODR, byly seřizovány pouze PID regulátorem. Neboť PID regulátor dostatečně reprezentuje funkčnost všech složek PID regulátoru, a postihnout všechny kombinace, které mohou při seřizování daných systémů nastat je nemožné.

A) ODR prvního řádu levé strany a nultého řádu pravé strany:

Regulátor typu P:

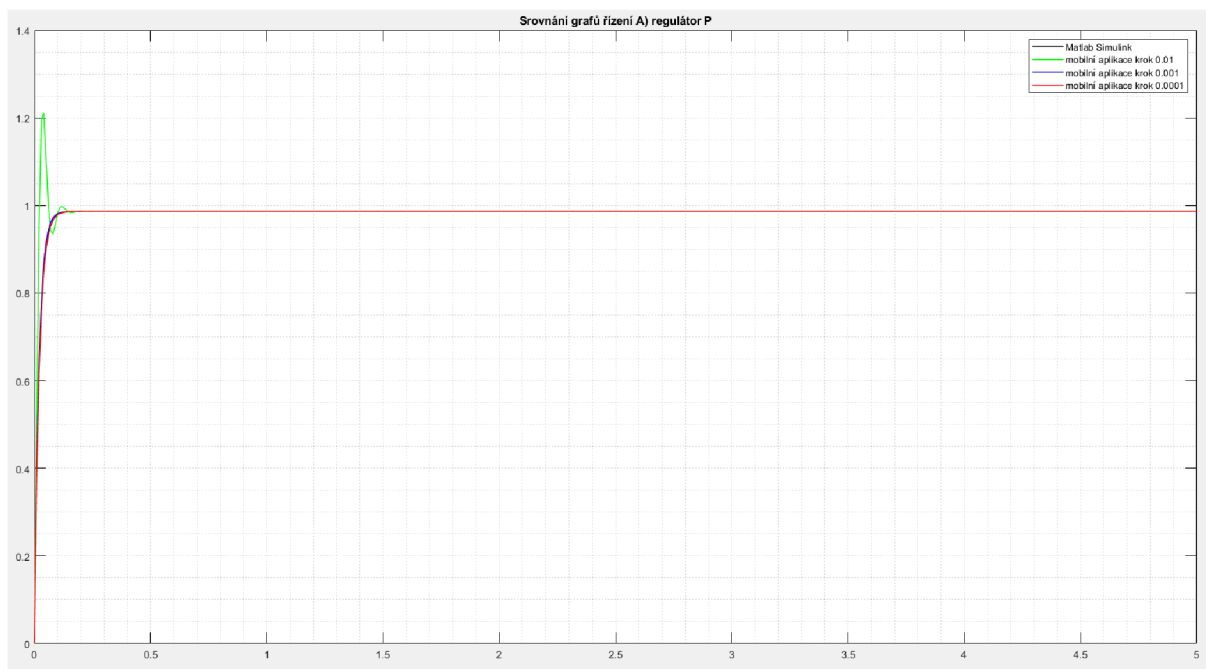


Obr. 4.52 Verifikace řízení A) regulátor typu P Matlab Simulink



Obr. 4.53 Verifikace řízení A) regulátor typu P mobilní aplikace s různou velikostí kroku zleva krok: 0.01, 0.001 a 0.0001

Grafické zobrazení výsledků verifikace řízení:



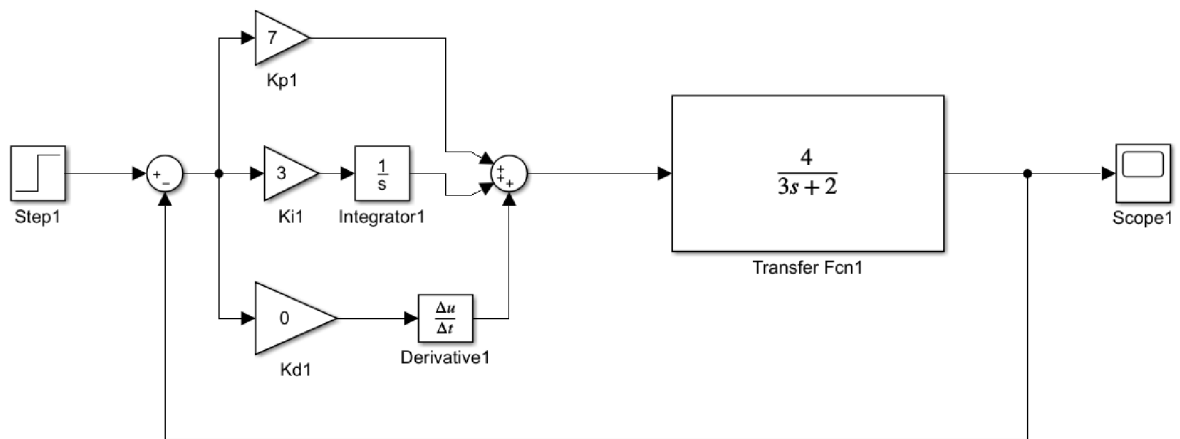
Obr. 4.54 Grafické zobrazení výsledků verifikace řízení A) regulátor typu P

Suma absolutních odchylek, od Matlab Simulink, při daném kroku integrace:

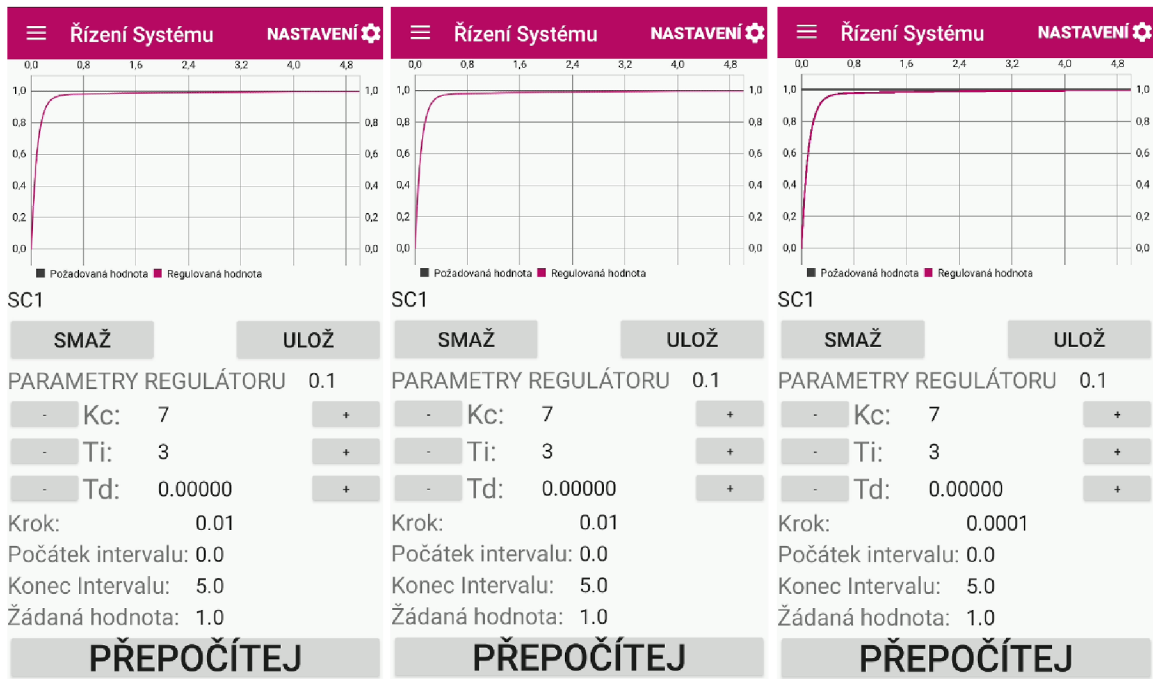
Krok	Suma absolutních odchylek
0.01	1.6304
0.001	0.1457
0.0001	0.0157

Tab. 4.13 Suma odchylek od Matlab Simulink A) regulátor P

Regulátor typu PI:

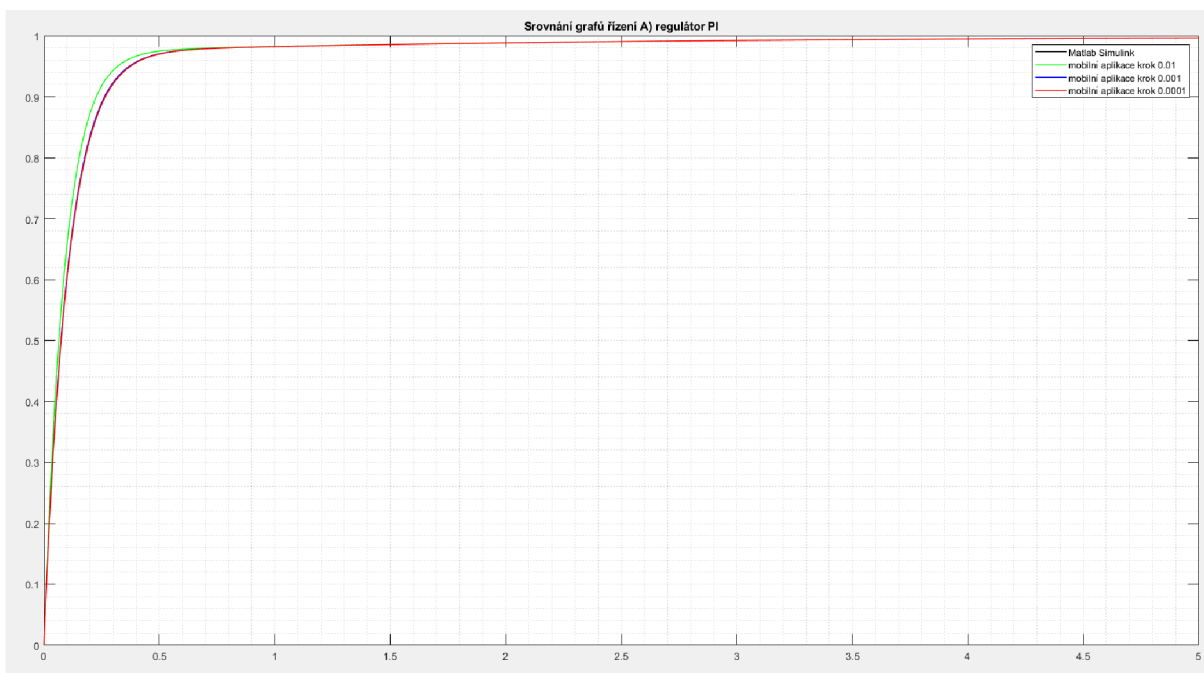


Obr. 4.55 Verifikace řízení A) regulátor typu PI Matlab Simulink



Obr. 4.56 Verifikace řízení A) regulátor typu PI mobilní aplikace s různou velikostí kroku zleva krok: 0.01, 0,001 a 0.0001

Grafické zobrazení výsledků verifikace řízení:



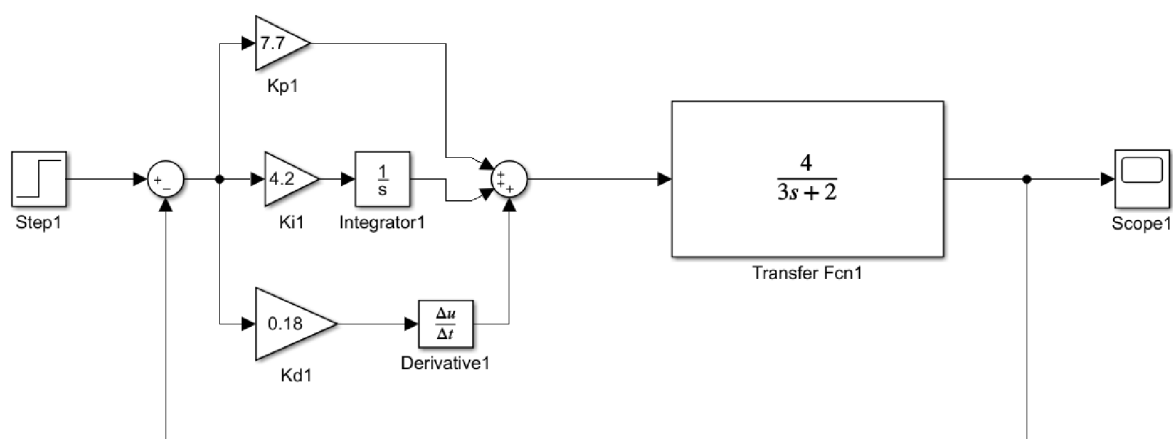
Obr. 4.57 Grafické zobrazení výsledků verifikace řízení A) regulátor typu PI

Suma absolutních odchylek, od Matlab Simulink, při daném kroku integrace:

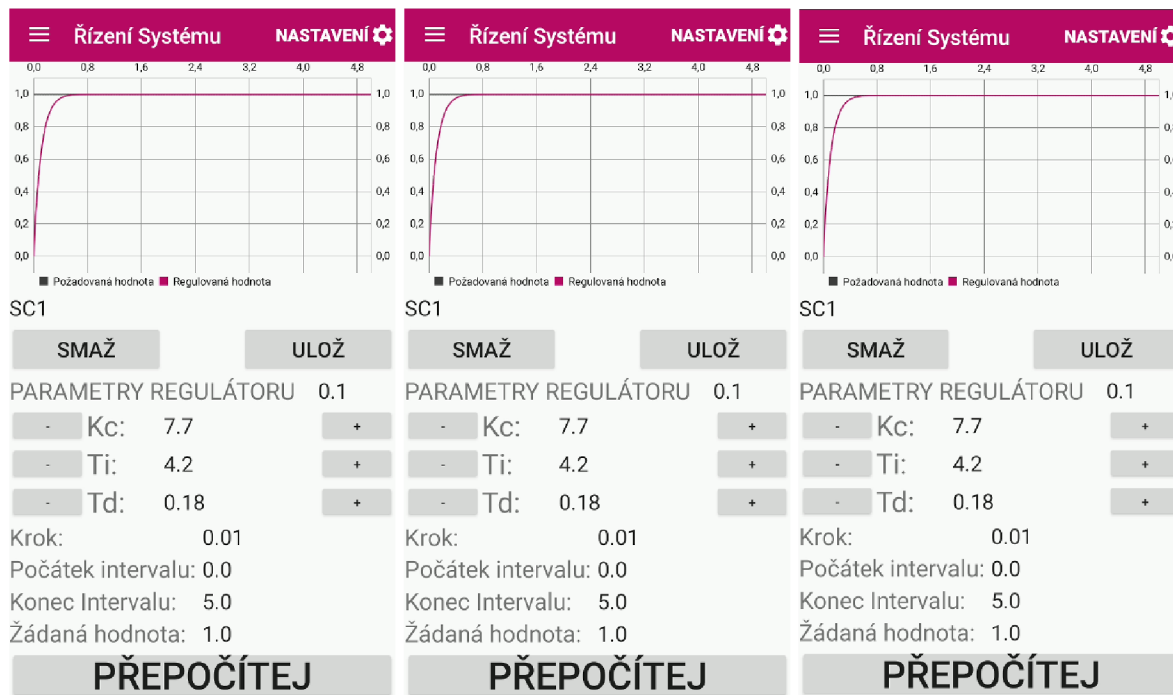
Krok	Suma absolutních odchylek
0.01	1.5036
0.001	0.1496
0.0001	0.0150

Tab. 4.14 Suma odchylek od Matlab Simulink A) regulátor PI

Regulátor typu PID:

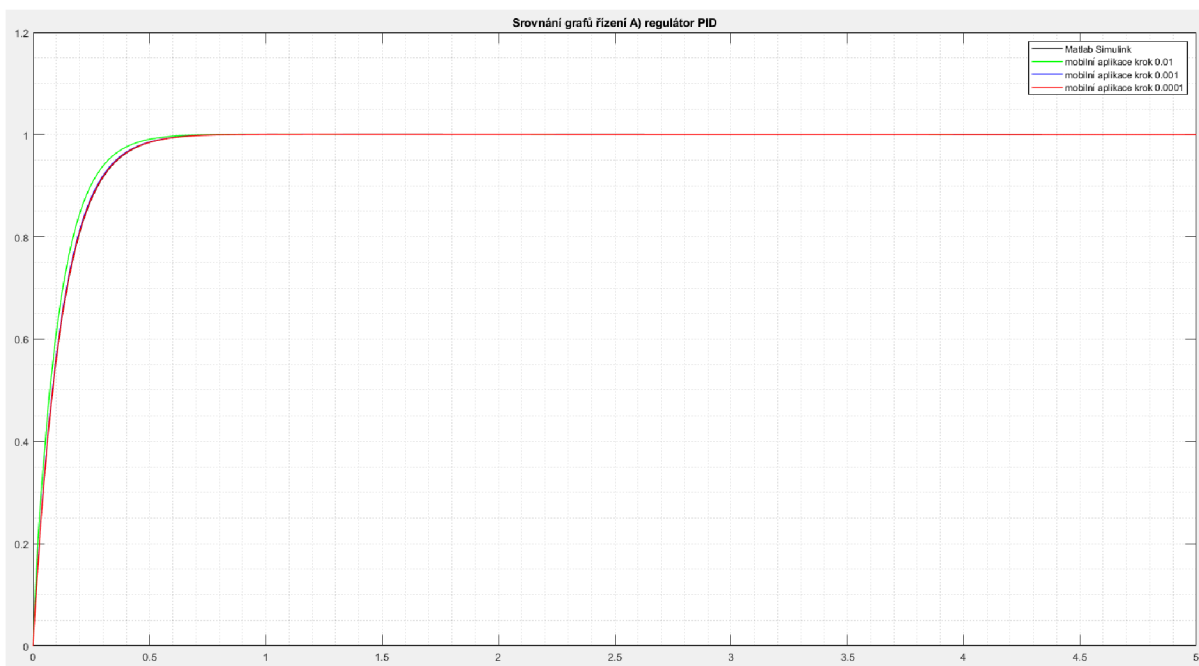


Obr. 4.58 Verifikace řízení A) regulátor typu PID Matlab Simulink



Obr. 4.59 Verifikace řízení A) regulátor typu PID mobilní aplikace s různou velikostí kroku zleva krok: 0.01, 0.001 a 0.0001

Grafické zobrazení výsledků verifikace řízení:



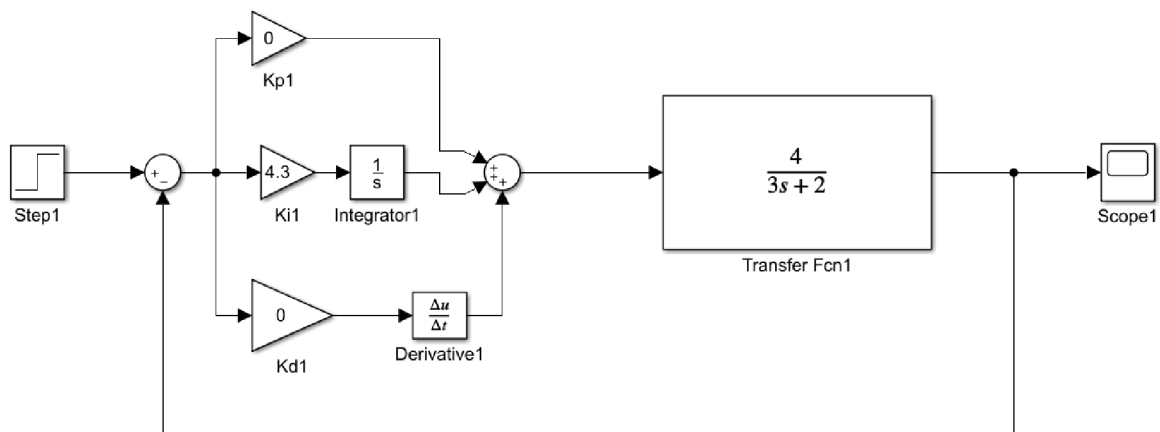
Obr. 4.60 Grafické zobrazení výsledků verifikace řízení A) regulátor typu PID

Suma absolutních odchylek, od Matlab Simulink, při daném kroku integrace:

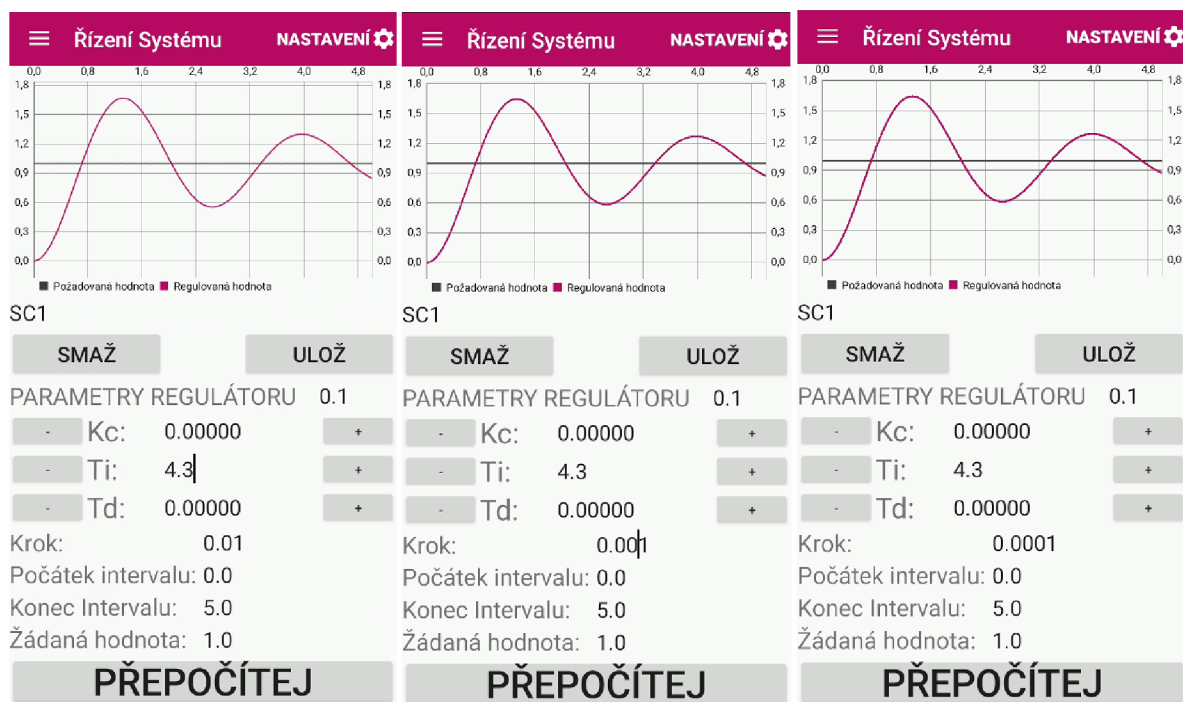
Krok	Suma absolutních odchylek
0.01	1.5669
0.001	0.2042
0.0001	0.0769

Tab. 4.15 Suma odchylek od Matlab Simulink A) regulátor PID

Regulátor typu I:



Obr. 4.61 Verifikace řízení A) regulátor typu I Matlab Simulink



Obr. 4.62 Verifikace řízení A) regulátor typu I mobilní aplikace s různou velikostí kroku zleva krok: 0.01, 0.001 a 0.0001

Grafické zobrazení výsledků verifikace řízení:



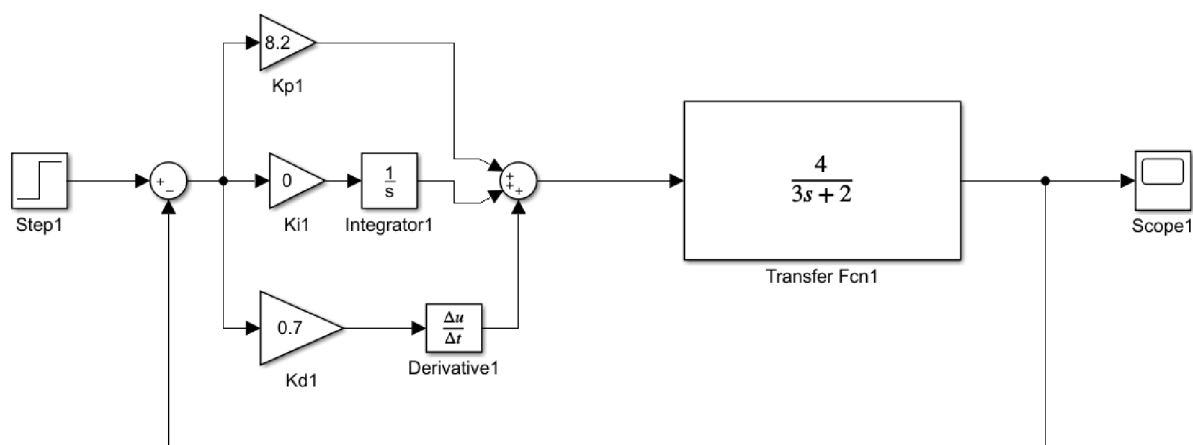
Obr. 4.63 Grafické zobrazení výsledků verifikace řízení A) regulátor typu I

Suma absolutních odchylek, od Matlab Simulink, při daném kroku integrace:

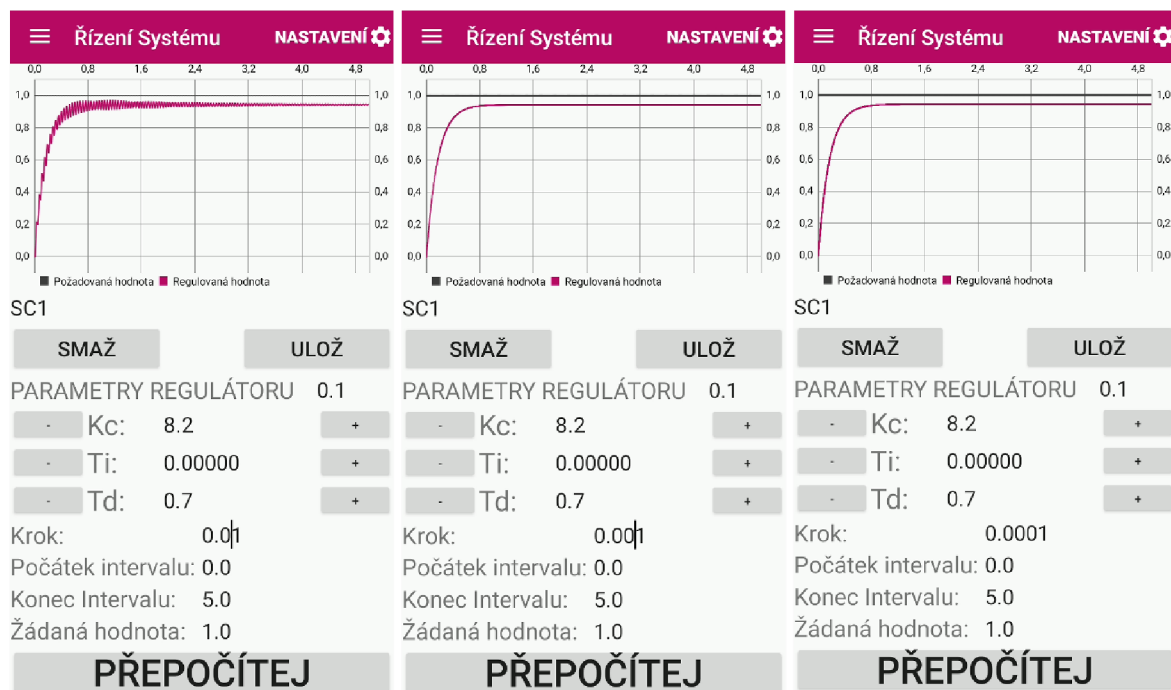
Krok	Suma absolutních odchylek
0.01	9.6432
0.001	0.9339
0.0001	0.0931

Tab. 4.16 Suma odchylek od Matlab Simulink A) regulátor I

Regulátor typu PD:

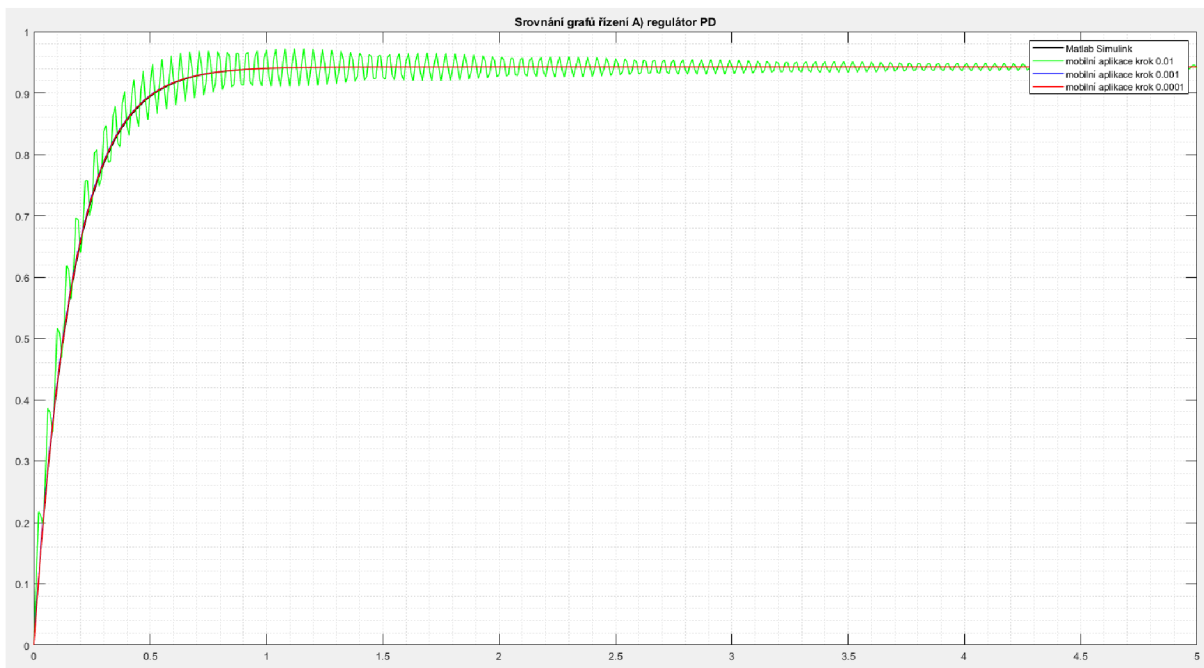


Obr. 4.64 Verifikace řízení A) regulátor typu PD Matlab Simulink



Obr. 4.65 Verifikace řízení A) regulátor typu PD mobilní aplikace s různou velikostí kroku zleva krok: 0.01, 0.001 a 0.0001

Grafické zobrazení výsledků verifikace řízení:



Obr. 4.66 Grafické zobrazení výsledků verifikace řízení A) regulátor typu PD

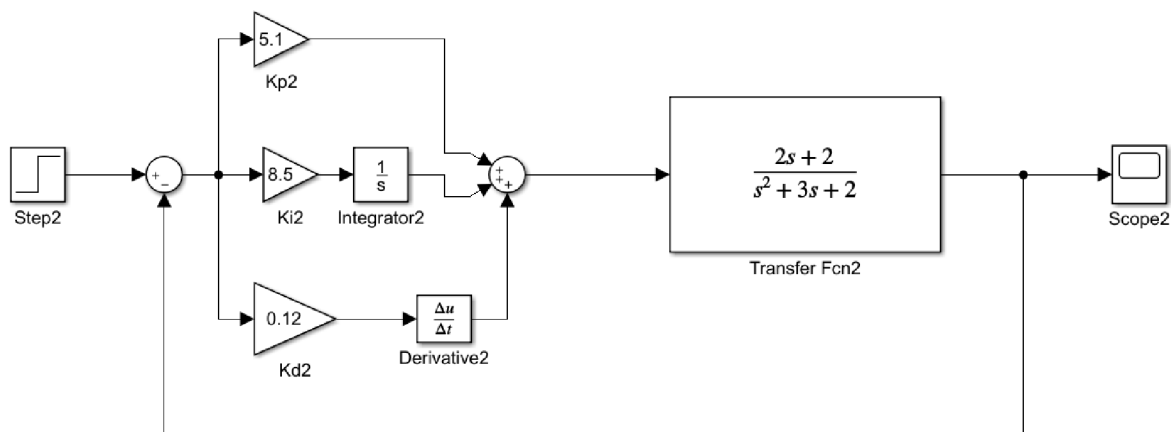
Suma absolutních odchylek, od Matlab Simulink, při daném kroku integrace:

Krok	Suma absolutních odchylek
0.01	6.6642
0.001	0.2951
0.0001	0.1753

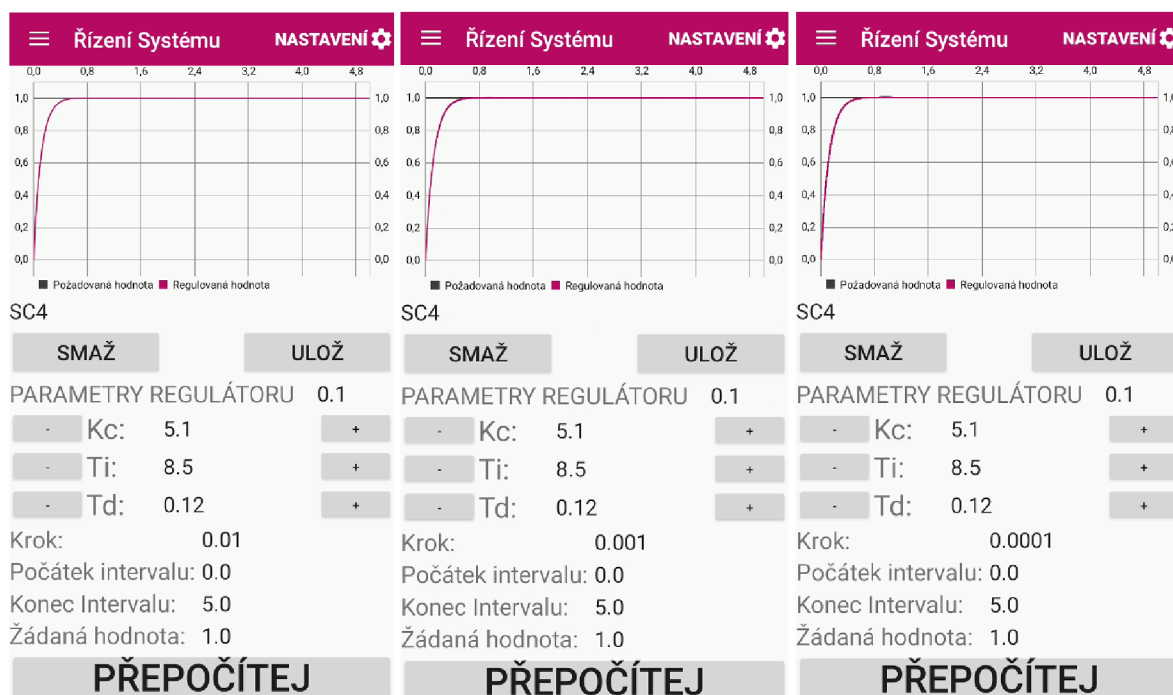
Tab. 4.17 Suma odchylek od Matlab Simulink A) regulátor PD

B) ODR druhého řádu levé strany a prvního řádu pravé strany:

Regulátor typu PID:

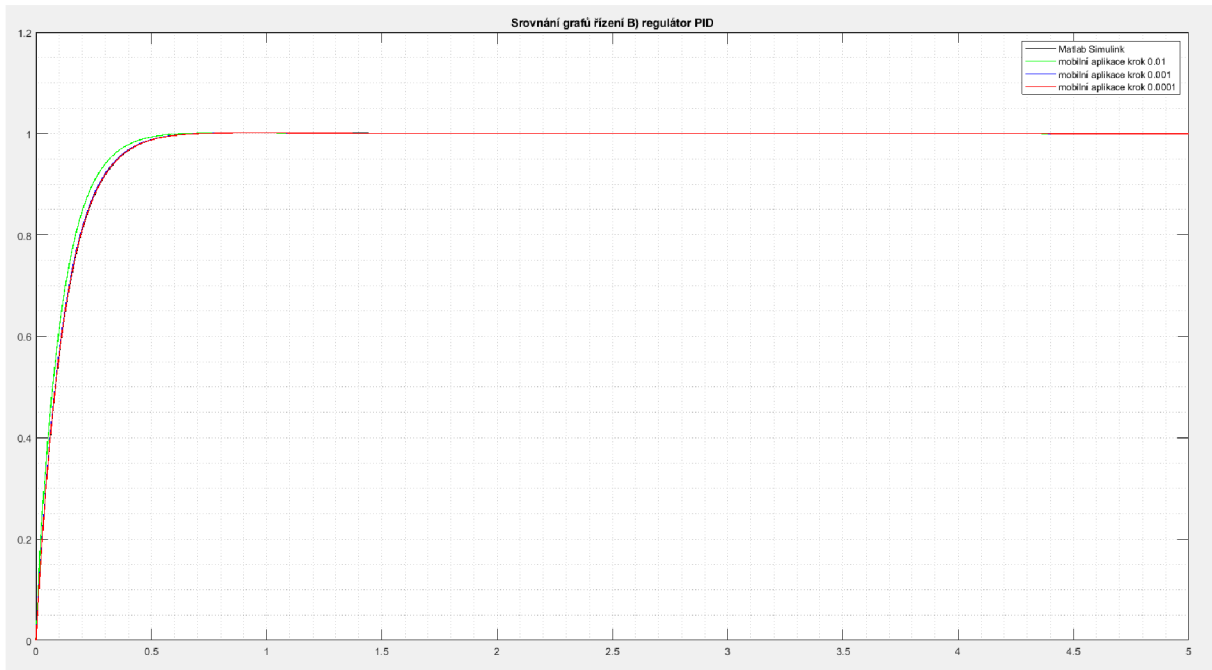


Obr. 4.67 Verifikace řízení B) regulátor typu PID Matlab Simulink



Obr. 4.68 Verifikace řízení B) regulátor typu PID mobilní aplikace s různou velikostí kroku zleva krok: 0.01, 0.001 a 0.0001

Grafické zobrazení výsledků verifikace řízení:



Obr. 4.69 Grafické zobrazení výsledků verifikace řízení B) regulátor typu PID

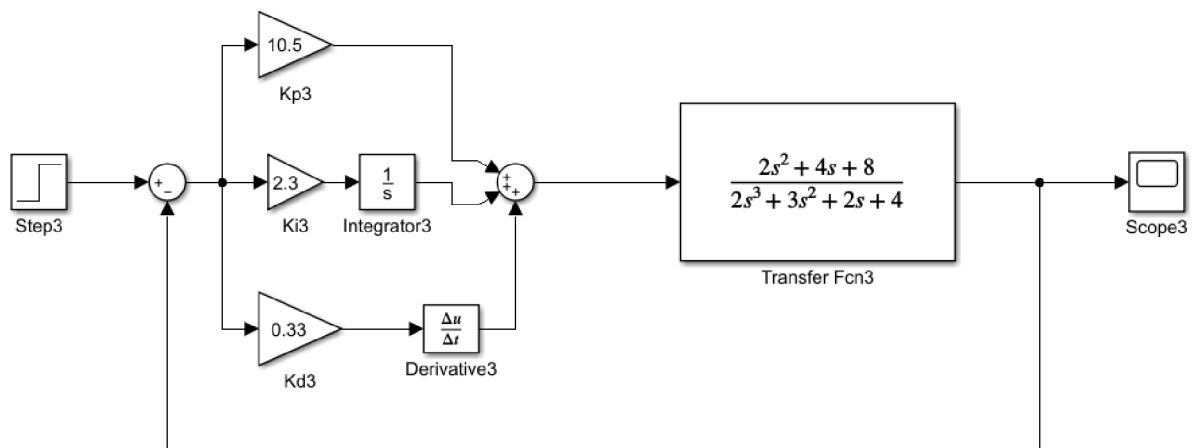
Suma absolutních odchylek, od Matlab Simulink, při daném kroku integrace:

Krok	Suma absolutních odchylek
0.01	1.5575
0.001	0.1892
0.0001	0.0649

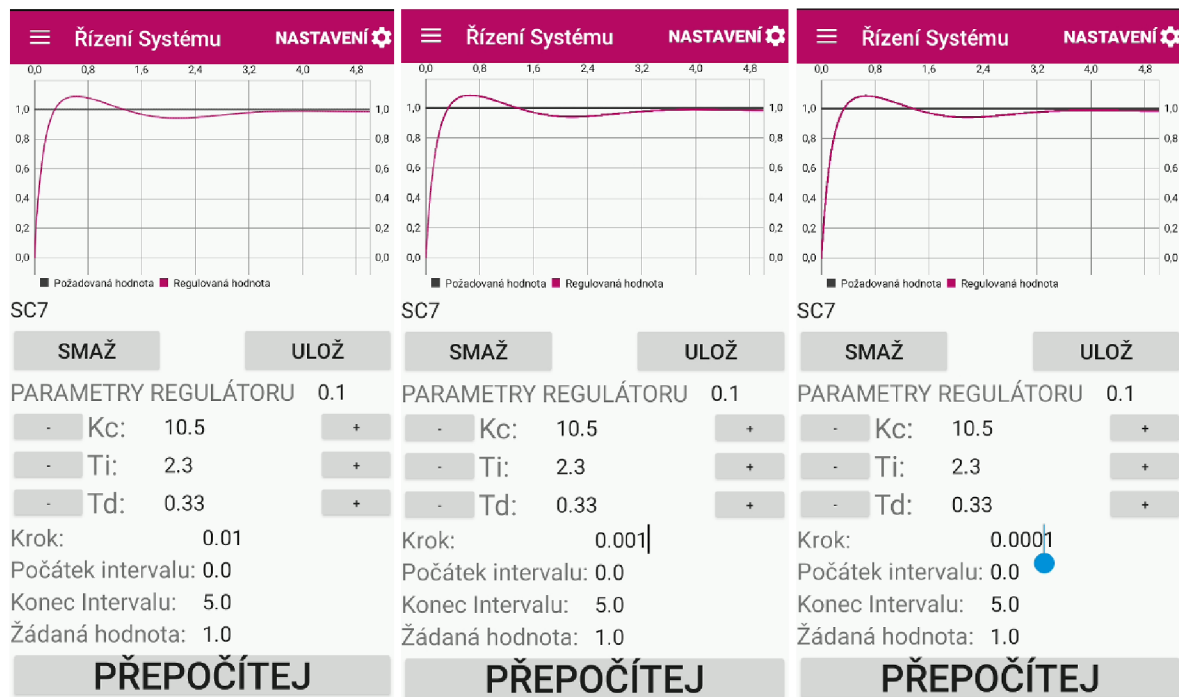
Tab. 4.18 Suma odchylek od Matlab Simulink B) regulátor PID

C) ODR třetího řádu levé strany a druhého řádu pravé strany:

Regulátor typu PID:

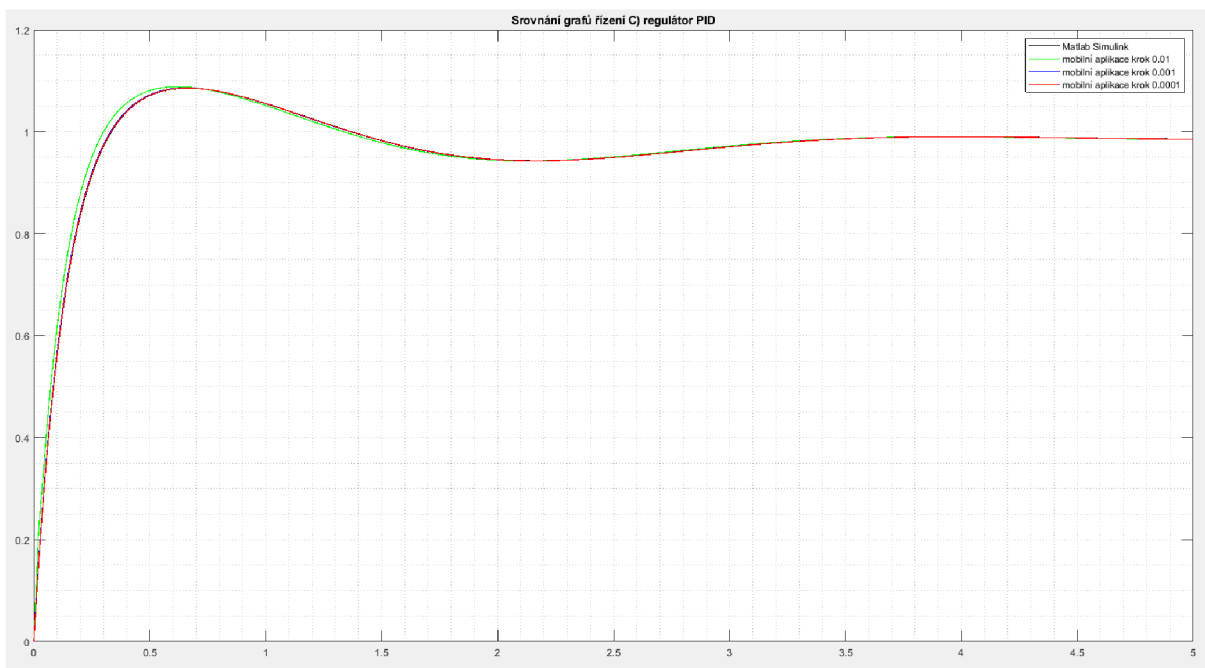


Obr. 4.70 Verifikace řízení C) regulátor typu PID Matlab Simulink



Obr. 4.71 Verifikace řízení C) regulátor typu PID mobilní aplikace s různou velikostí kroku zleva krok: 0.01, 0.001 a 0.0001

Grafické zobrazení výsledků verifikace řízení:



Obr. 4.72 Grafické zobrazení výsledků verifikace řízení C) regulátor typu PID

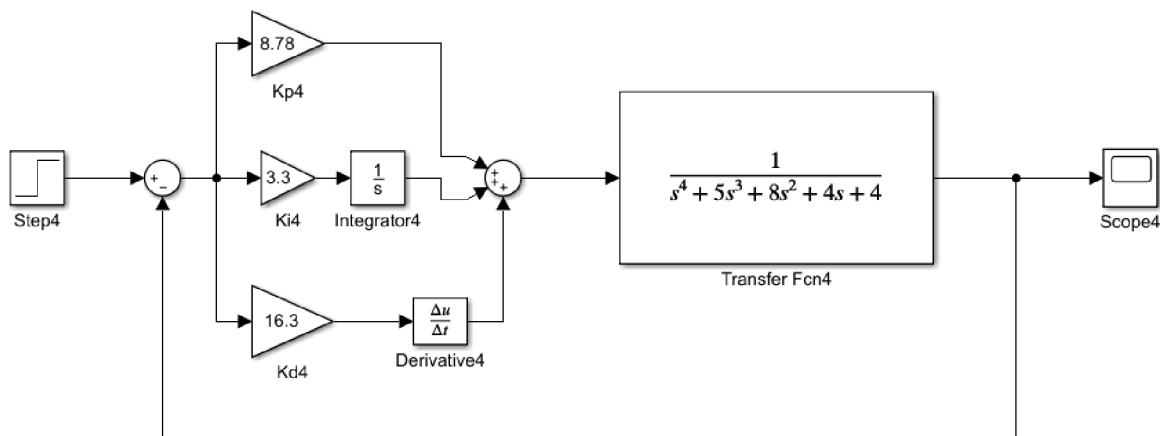
Suma absolutních odchylek, od Matlab Simulink, při daném kroku integrace:

Krok	Suma absolutních odchylek
0.01	2.4622
0.001	0.3075
0.0001	0.1088

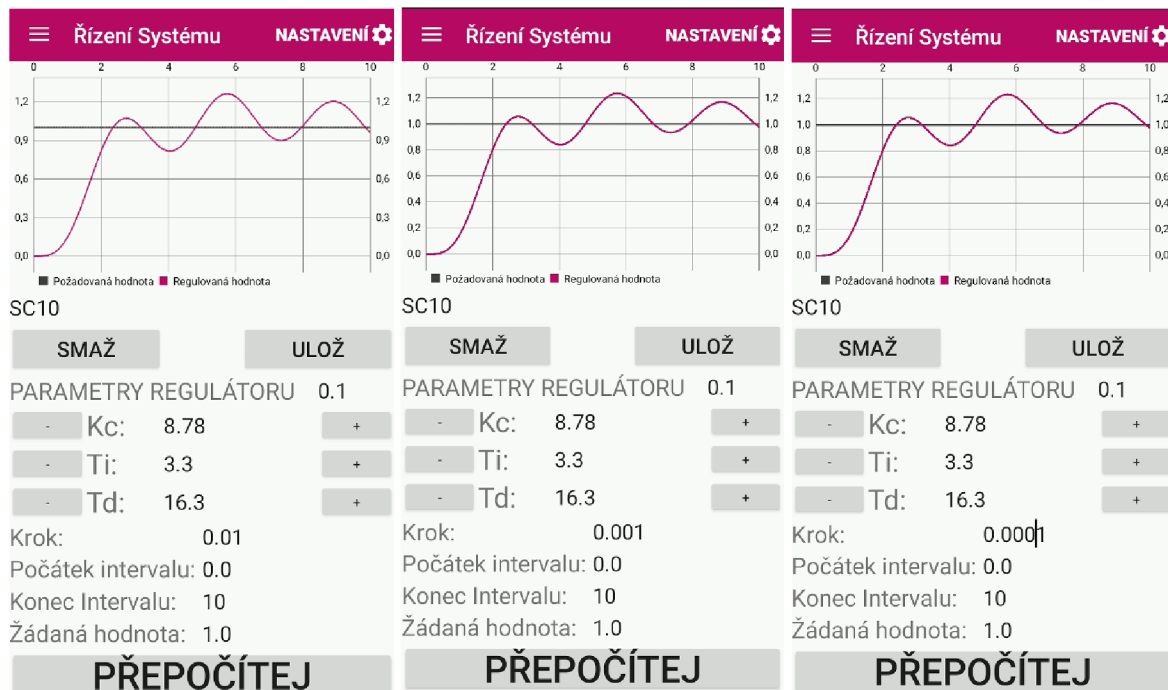
Tab. 4.19 Suma odchylek od Matlab Simulink C) regulátor PID

D) ODR čtvrtého řádu levé strany a nultého řádu pravé strany:

Regulátor typu PID:

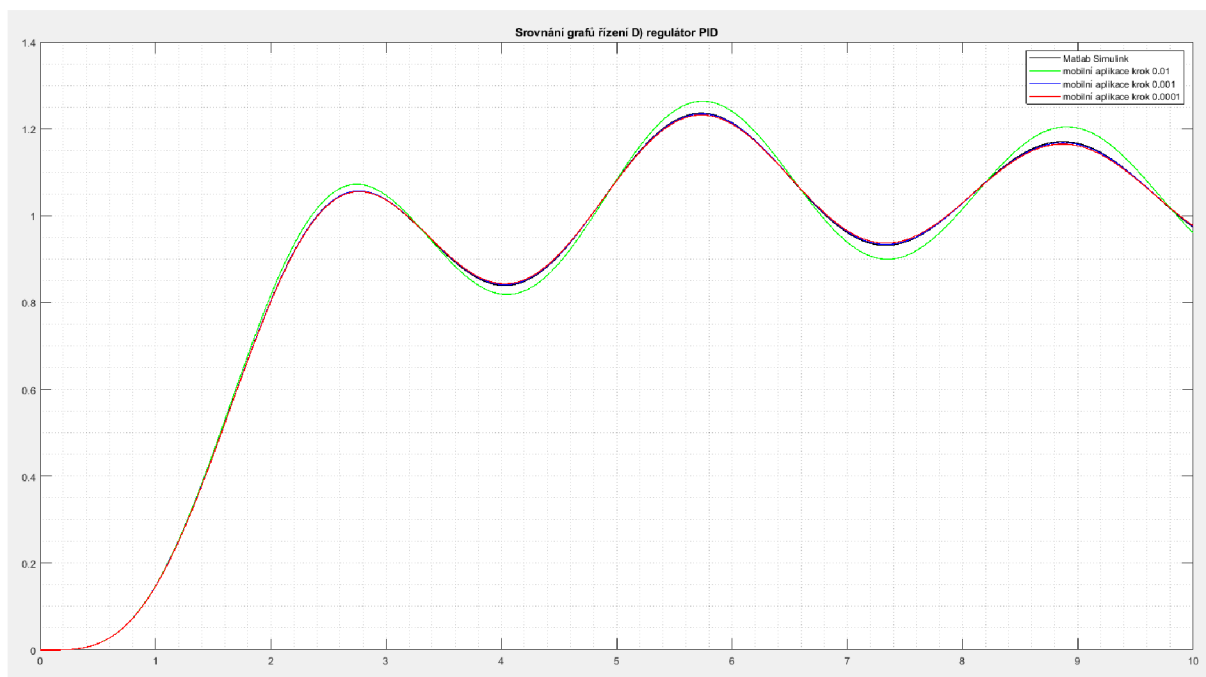


Obr. 4.73 Verifikace řízení D) regulátor typu PID Matlab Simulink



Obr. 4.74 Verifikace řízení D) regulátor typu PID mobilní aplikace s různou velikostí kroku zleva krok: 0.01, 0.001 a 0.0001

Grafické zobrazení výsledků verifikace řízení:



Obr. 4.75 Grafické zobrazení výsledků verifikace řízení D) regulátor typu PID

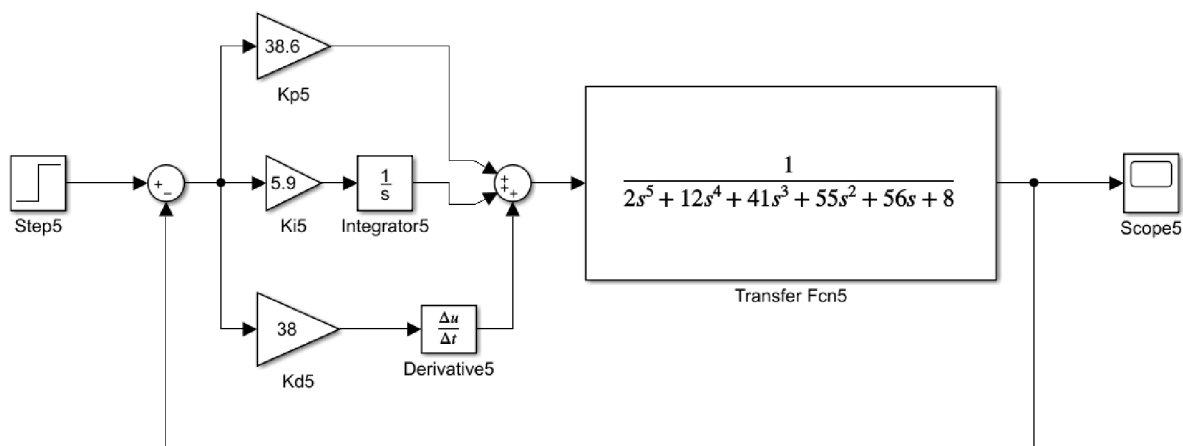
Suma absolutních odchylek, od Matlab Simulink, při daném kroku integrace:

Krok	Suma absolutních odchylek
0.01	14.5060
0.001	1.0737
0.0001	2.3568

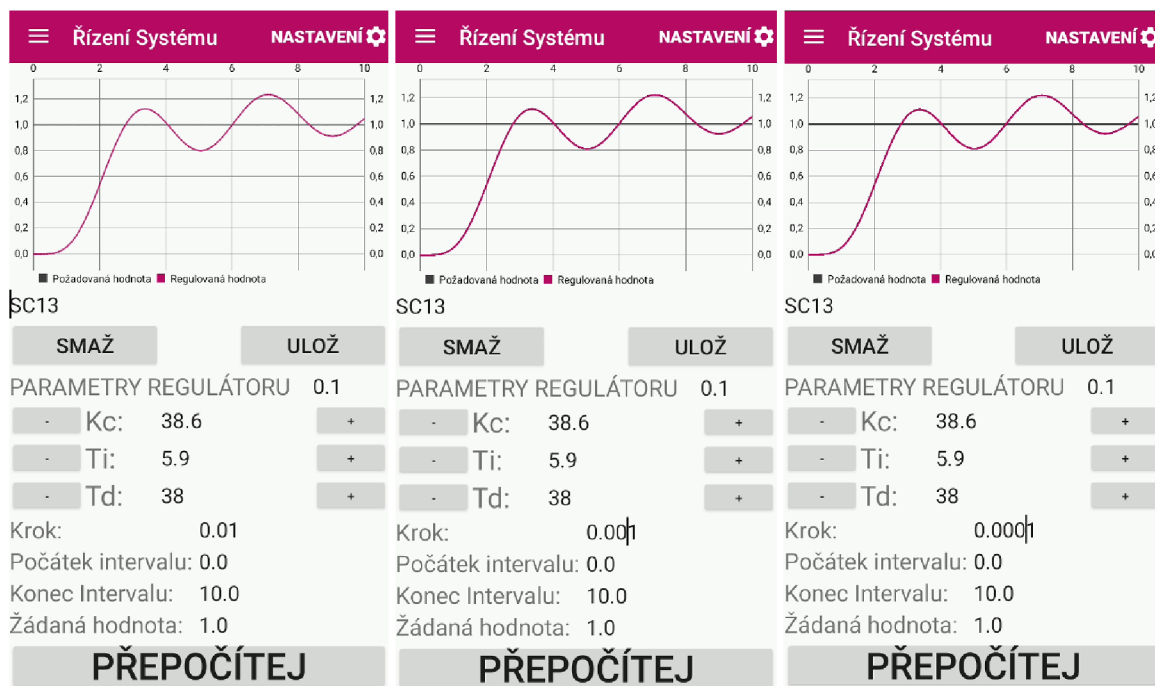
Tab. 4.20 Suma odchylek od Matlab Simulink D) regulátor PID

E) ODR pátého řádu levé strany a nultého řádu pravé strany:

Regulátor typu PID:

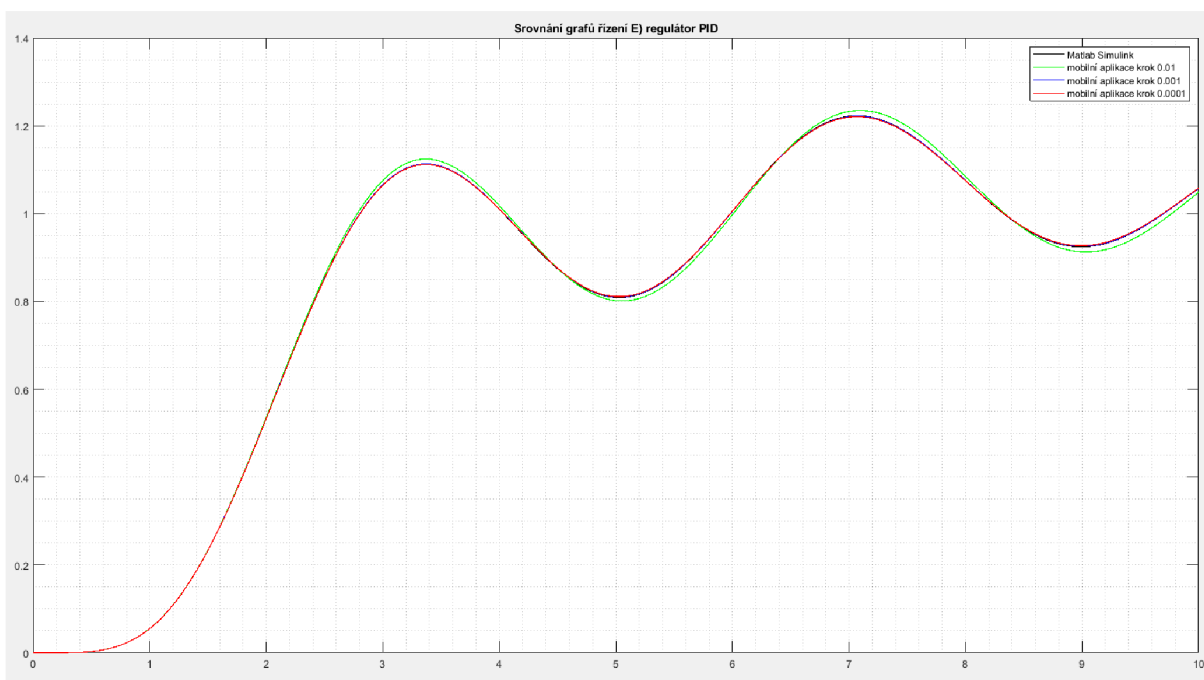


Obr. 4.76 Verifikace řízení E) regulátor typu PID Matlab Simulink



Obr. 4.77 Verifikace řízení E) regulátor typu PID mobilní aplikace s různou velikostí kroku zleva krok: 0.01, 0.001 a 0.0001

Grafické zobrazení výsledků verifikace řízení:



Obr. 4.78 Grafické zobrazení výsledků verifikace řízení E) regulátor typu PID

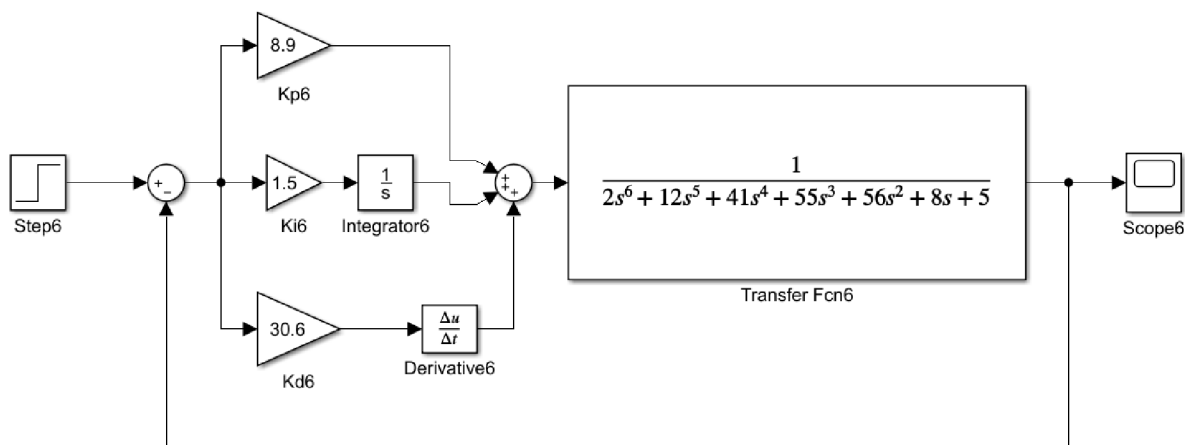
Suma absolutních odchylek, od Matlab Simulink, při daném kroku integrace:

Krok	Suma absolutních odchylek
0.01	6.9389
0.001	0.5101
0.0001	0.9765

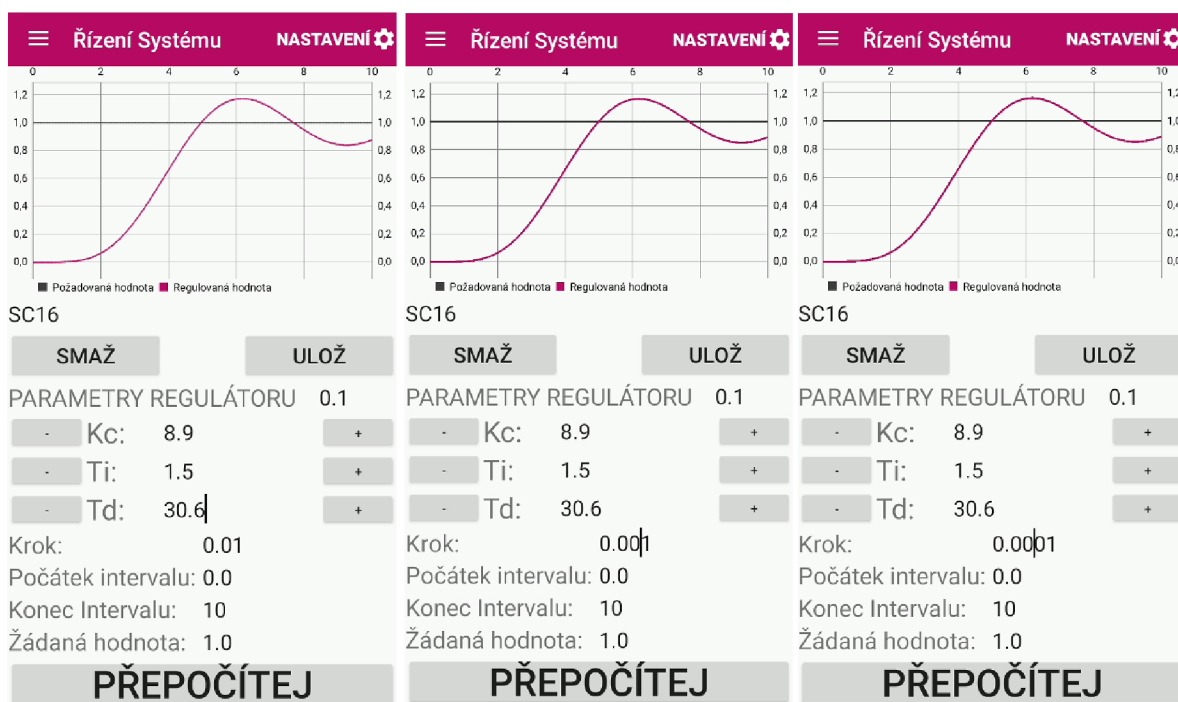
Tab. 4.21 Suma odchylek od Matlab Simulink E) regulátor PID

F) ODR šestého řádu levé strany a nultého řádu pravé strany:

Regulátor typu PID:

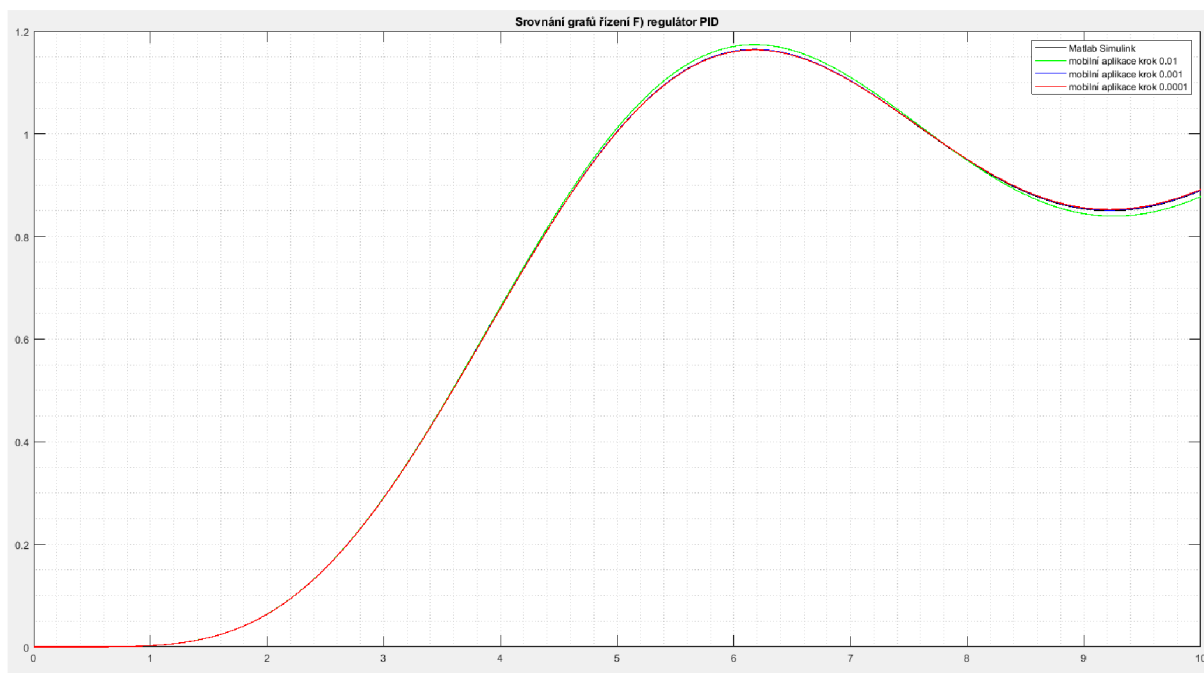


Obr. 4.79 Verifikace řízení F) regulátor typu PID Matlab Simulink



Obr. 4.80 Verifikace řízení F) regulátor typu PID mobilní aplikace s různou velikostí kroku zleva krok: 0.01, 0.001 a 0.0001

Grafické zobrazení výsledků verifikace řízení:



Obr. 4.81 Grafické zobrazení výsledků verifikace řízení F) regulátor typu PID

Suma absolutních odchylek, od Matlab Simulink, při daném kroku integrace:

Krok	Suma absolutních odchylek
0.01	4.3949
0.001	0.3261
0.0001	0.7077

Tab. 4.22 Suma odchylek od Matlab Simulink F) regulátor PID

4.3.1 Shrnutí verifikace seřizování PID regulátoru:

Data spočtená mobilní aplikací, a Matlab Simulink, se liší při použití kroku integrace o velikosti 0.001 pouze nepatrně. Prezentovaná přesnost, je více než dostačující, pro praktické využití aplikace. Obecně se dá říci, že kvůli implementovanému výpočtovému mechanismu v aplikaci (viz. 3.3), se přesnost výpočtu zvyšuje, se zmenšením výpočtového integračního kroku. Praktické pokusy ukázaly, platnost tohoto tvrzení bez D složky. Pokud je v regulátoru přídavná D složka, může nadměrné snižování výpočtového kroku vést naopak ke zhoršení výsledků viz. výše. Velikost kroku by měla být volena, s ohledem, na velikost časového horizontu, žádanou přesnost, a výpočetní možnosti daného zařízení.

ZÁVĚR

V rámci zpracování této diplomové práce, byl vytvořen software (aplikace) pro mobilní zařízení s OS Android (APK 14-30): APLIKACE PRO VÝPOČET A SEŘÍZENÍ PID REGULÁTORU PRO ZAŘÍZENÍ S OPERAČNÍM SYSTÉMEM ANDROID.

Aplikace je schopna vytvořit matematický model (identifikovat) dynamického systému, na základě dat odezvy systému na jednotkový skok, a dále je schopná takto identifikovaný model využít, k seřízení a optimálnímu nastavení PID regulátoru.

Jako jádro výpočtového aparátu aplikace využívá metod numerické optimalizace (Metoda flexibilního simplexu) a numerické integrace (Metoda Runge Kutta čtvrtého řádu).

Správnost vypočtených výsledků byla zkontrolována certifikovaným výpočtovým softwarem MATLAB ver. 2019.

Implementace v prostředí OS android je provedeno, s ohledem na ergonomii a komfort uživatele, a celý proces zpracování naměřeného signálu, je logicky rozčleněn do snadno pochopitelných částí.

Celý program byl sestaven ve freewarovém IDE Android Studio, za pomoci programovacího jazyka JAVA.

Hlavní přínos této práce spočívá, v dosavadní neexistenci obdobného softwaru na dané platformě, a možnostech dalšího vývoje tohoto softwaru, ku příkladu: implementace dalších metod numerické integrace a optimalizace, vylepšení systému načítání a ukládání dat, implementace diskrétní identifikace systémů, možnost online identifikace systému a mnoho dalších.

POUŽITÁ LITERATŮRA A ZDROJE:

LITERATURA:

- [1] BALÁTĚ, Jaroslav. Automatické řízení. Praha: BEN - technická literatura, 2003. ISBN 978-80-7300-020-2.
- [2] DARWIN, Ian F. *Java: kuchařka programátora: [vzory a řešení pro vaše aplikace]*. Brno: Computer Press, 2006. ISBN 80-251-0944-5.
- [3] O'DWYER, Aidan. *Handbook of PI and PID Controller Tuning Rules*, 3rd ed. edition 2009. IMPERIAL COLLEGE PRESS, 2009. ISBN 978-1848162426.
- [4] HOFREITER, Milan. *Základy automatického řízení*. V Praze: České vysoké učení technické, 2012. ISBN 978-80-01-05007-1.
- [5] LACKO, Ľuboslav. *Vývoj aplikací pro Android*. Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.
- [6] SCHILDT, Herbert. *Java 7: výukový kurz*. Brno: Computer Press, 2012. ISBN 978-80-251-3748-2.
- [7] ISERMANN, Rolf a Marco MÜNCHHOF. *Identification of dynamic systems: an introduction with applications*. Berlin: Springer, 2011. ISBN 978-3-540-78878-2.
- [8] KEESMAN, Karel J. *System Identification*. London: Springer London, 2011. Advanced Textbooks in Control and Signal Processing. ISBN 978-0-85729-521-7
- [9] BALÓDY, O. *Metody Runge-Kutta*. Olomouc, 2011. Bakalářská práce. Univerzita Palackého v Olomouci. Vedoucí práce Jitka Machalová.
- [10] Gao, F., Han, L. Implementing the Nelder-Mead simplex algorithm with adaptive parameters. *Comput Optim Appl* 51, 259–277 (2012). <https://doi.org/10.1007/s10589-010-9329-3>

ZDROJE DOSTUPNÉ ONLINE:

- [12] Android Tutorial - Tutlane. Tutorials for Asp.net MVC, LINQ, C#, SQL Server, Android, IOS, AngularJS, JOOMLA, JAVA, Perl, Php, MySQL - Tutlane - Tutlane [online]. Dostupné z: <https://www.tutlane.com/tutorial/android>
- [13] Stack Overflow - Where Developers Learn, Share, & Build Careers. Stack Overflow - Where Developers Learn, Share, & Build Careers [online]. Dostupné z: <https://stackoverflow.com>
- [14] shad sluiteer - YouTube. YouTube [online]. Copyright © 2021 Google LLC [cit. 04.04.2021]. Dostupné z: <https://www.youtube.com/channel/UCUSqKFDbRaaWQTaMN3r4Pbg>

ZÁPISKY A POZNATKY Z PŘEDMĚTŮ:

- [15] MOUČKA, M. *Aplikovaná Kybernetika*. Technická Univerzita v Liberci, 2017.
- [16] GARAN M, *Simulace a identifikace systémů*. Technická Univerzita v Liberci, 2020.
- [17] VOTRUBEC, R. *Teorie automatického řízení*. Technická Univerzita v Liberci, 2019.