VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

SOUBOR DEMONSTRAČNÍCH ÚLOH PRO PŘEDMĚT ÚVOD DO KYBERNETIKY

A SET OF TASKS FOR THE COURSE OF INTRODUCTION TO CYBERNETICS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

Markéta Jalůvková

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Miroslav Jirgl, Ph.D.

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Studentka: Markéta Jalůvková ***ID:*** 211425

Ročník: 3 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Soubor demonstračních úloh pro předmět Úvod do kybernetiky

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout a implementovat sadu úloh pro demonstraci vlastností základních dynamických systémů pro možnost jejich využití v předmětu Úvod do kybernetiky.

1. Vyberte sadu reálných systémů vhodných pro demonstraci základních vlastností dynamických systémů a popište jejich zjednodušené matematické modely.
2. Implementujte modely v prostředí MATLAB/Simulink a proveďte základní simulace.
3. Seznamte se s tvorbou GUI v prostředí MATLAB a s možností propojení aplikace s modely realizovanými v Simulinku.
4. Navrhněte aplikaci (včetně GUI) pro demonstraci základních vlastností vybraných systémů.
5. Implementujte aplikaci v MATLAB App Designer.
6. Otestujte aplikaci a demonstrujte její funkčnost.

DOPORUČENÁ LITERATURA:

OPPENHEIM, Alan, WILLSKY, Alan. Signals and Systems. Second edition. New Jersey: Prentice Hall 1997, 957 s. ISBN 0-13-814757-4.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 23.5.2022

Vedoucí práce: Ing. Miroslav Jirgl, Ph.D.

doc. Ing. Václav Jirsík, CSc.
předseda rady studijního programu

Abstrakt

Bakalářská práce se zabývá tvorbou aplikací v prostředí MATLAB App Designer. Popisuje využití reálných systémů a základních dynamických vlastností v aplikacích GUI pro výuku studentů v předmětu Úvod do kybernetiky. Výsledkem práce je aplikace s vazbou na modely vybraných reálných systémů vytvořenými v prostředí MATLAB Simulink.

Klíčová slova

MATLAB, App Designer, Simulink, simulace, GUI

Abstract

This bachelor's thesis is about making application in MATLAB App Designer. It describe using real systems and basic dynamic properties in GUI applications. It has use for students of Entrance for kyberneticts. Goal is to make application with conection on chosen systems created in MATLAB Simulink interface.

Keywords

MATLAB, App Designer, Simulink, simulation, GUI

Bibliografická citace

JALŮVKOVÁ, Markéta. *Soubor demonstračních úloh pro předmět Úvod do kybernetiky*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřící techniky. Vedoucí práce: Ing. Miroslav Jirgl, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	Markéta Jalůvková
VUT ID studenta:	211425
Typ práce:	Bakalářská práce
Akademický rok:	2021/22
Téma závěrečné práce:	Soubor demonstračních úloh pro předmět Úvod do kybernetiky

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

.

V Brně dne: 12. května 2022 ----------

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Miroslavu Jirglovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce. Dále bych chtěla poděkovat Jakubu Vondrovi, mé rodině a Lydii Molíkové za podporu a pevné nervy, které semnou měli během psaní bakalářské práce. Závěrem bych chtěla poděkovat Mgr. Martině Doležalové za gramatickou korekci.

V Brně dne: 12. května 2022 ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ **8**

ÚVOD **9**

1. VÝBĚR REÁLNÝCH SYSTÉMŮ **10**

1.1 SYSTÉMY 1. ŘÁDU 10

1.1.1 Hydraulický systém *11*

1.1.2 Tepelný systém *13*

1.1.3 Elektrický systém 1. řádu *14*

1.2 SYSTÉMY 2. ŘÁDU 15

1.2.1 Elektrický systém 2. řádu *16*

2. IMPLEMENTACE MODELŮ VYBRANÝCH SYSTÉMŮ V PROSTŘEDÍ MATLAB SIMULINK **18**

2.1 IMPLEMENTACE A OVĚŘENÍ SYSTÉMU 1. ŘÁDU 18

2.1.1 Implementace hydraulického systému *18*

2.1.2 Implementace tepelného systému *22*

2.1.3 Implementace elektrického systému 1. řádu *24*

2.1.4 Ověření časových konstant simulovaných systému 1. řádu *25*

2.2 IMPLEMENTACE SYSTÉMU 2. ŘÁDU 28

3. GUI V PROSTŘEDÍ MATLAB **31**

3.1 TVORBA APP DESIGNER 31

3.2 PROPOJENÍ PROSTŘEDÍ SIMULINK A APP DESIGNER 31

4. NÁVRH APLIKACÍ PRO PROSTŘEDÍ APP DESIGNER **34**

4.1 NÁVRH ROZLOŽENÍ KOMPONENT V APLIKACÍCH 35

5. IMPLEMENTACE APLIKACÍ V PROSTŘEDÍ MATLAB APP DESIGNER **38**

5.1 IMPLEMENTACE ÚLOHY 1 38

5.2 IMPLEMENTACE ÚLOHY 2 41

5.3 IMPLEMENTACE ÚLOHY 3 41

5.4 IMPLEMENTACE ÚLOHY 4 42

6. DEMONSTRACE FUNKČNOSTI APLIKACÍ **44**

6.1 TEST APLIKACE ÚLOHY 1 44

6.2 TEST APLIKACE ÚLOHY 2 45

6.3 TEST APLIKACE ÚLOHY 3 46

6.4 TEST APLIKACE ÚLOHY 4 47

ZÁVĚR **49**

LITERATURA **50**

SEZNAM PŘÍLOH **51**

SEZNAM OBRÁZKŮ

Obrázek 1.1.1: Model hydraulické soustavy 11
Obrázek 1.1.2: Odezva systému na jednotkový skok 11
Obrázek 1.1.3: Tepelný systém 1. řádu 13
Obrázek 1.1.4: Schéma zapojení RC článku 14
Obrázek 1.2.1: Přechodová charakteristika systému 2. řádu 15
Obrázek 1.2.2: Schéma zapojené RLC obvodu 16
Obrázek 2.1.1: Schéma nelineárního systému MATLAB Simulink 19
Obrázek 2.1.2: Schéma parametrizovaného systému MATLAB Simulink 19
Obrázek 2.1.3: Závislost výšky hladiny na čase nelineárního a parametrizovaného systému 20
Obrázek 2.1.4 Schéma linearizovaného hydraulického systému 20
Obrázek 2.1.5: Závislosti výšky hladina na čase linearizovaného hydraulického systému 21
Obrázek 2.1.6: Schéma linearizovaného tepelného systému 22
Obrázek 2.1.7: Graf závislosti teploty kapaliny na čase omezený bodem varu 23
Obrázek 2.1.8: Graf závislosti teploty kapaliny na čase linearizovaného tepelného systému 23
Obrázek 2.1.9: Schéma linearizovaného RC článku 24
Obrázek 2.1.10: Graf závislosti výstupního napětí na čase RC článku 24
Obrázek 2.1.11: Reakce systému na jednotkový skok [2] 25
Obrázek 2.1.12: Grafické odečtení časové konstanty hydraulického systému 27
Obrázek 2.1.13: Grafické odečtení časové konstanty tepelného systému 27
Obrázek 2.1.14: Grafické odečtení časové konstanty elektrického systému 28
Obrázek 2.2.1: Schéma linearizovaného elektrického systému 2. řádu 29
Obrázek 2.2.2: Graf závislosti výstupního napětí na čase RLC systému 30
Obrázek 3.2.1: Principiální propojení aplikace App Designer a modelu Simulink 32
Obrázek 4.1.1: Návrh GUI aplikace – Úloha 2 35
Obrázek 4.1.2: Návrh GUI aplikace – Úloha 1 36
Obrázek 4.1.3: Návrh GUI aplikace – Úloha 4 37
Obrázek 4.1.4: Návrh GUI aplikace – Úloha 3 37
Obrázek 6.1.1: Test aplikace Úlohy 1 45
Obrázek 6.2.1: Test aplikace Úlohy 2 46
Obrázek 6.3.1: Test aplikace Úlohy 3 47
Obrázek 6.4.1: Test aplikace Úlohy 4 48

Úvod

Cílem bakalářské práce je navrhnout a vytvořit aplikace k jednoduchým reálným systémům, se kterými se v praxi i v běžném životě setkáváme nejčastěji. Aplikace se následně můžou využit jako učební pomůcka pro studenty předmětu Úvod do kybernetiky k demonstraci základních dynamických vlastností systémů.

V první kapitole se čtenář seznámí s jednotlivými systému, jejich dynamickými vlastnosti a vyjádřením v diferenciální rovnici. Následně systémy implementujeme a provedeme simulace v prostředí MATLAB Simulink. V kapitole třetí přiblížíme využití a tvorbu GUI aplikací ve vývojovém prostředí MATLAB spolu možností propojení vývojových prostředí Simulink a App Designer. V prostředí App Designer již zůstaneme a navrhneme vizuál jednotlivých aplikací pro demonstraci vlastností systémů. V předposlední kapitole napíšeme kód, který utváří chování aplikací na akce jejich uživatelů. A nakonec otestujeme, zda implementované GUI aplikací pracují správně a bez nechtěných varování či chyb při přímém spuštění v prostředí MATLAB.

1. VÝBĚR REÁLNÝCH SYSTÉMŮ

Výběr reálných systémů byl proveden s ohledem na možnosti demonstrovat a analyzovat základní dynamické vlastnosti těchto systémů zejména z průběhu přechodových charakteristik. Mezi tyto vlastnosti patří:

- Řád systému
- Linearita a její důsledky
- Velikost časové konstanty
- Kmitavost odezvy

Jednotlivé vlastnosti v následujících podkapitolách zastupuje reálný hydraulický systém, tepelný systém a elektrické systémy 1. a 2. řádu.

1.1 Systémy 1. řádu

Tato kapitola porovnává tři různé fyzikální systémy prvního řádu a jejich časové konstanty, které lze snadno určit například z operátorového přenosu *F(p)*. Operátorový přenos získáme předením diferenciální rovnice na polynomickou a pomocí Laplaceovy transformace dostaneme předpis *F(p)* [1].

$$F(p) = L\{f(t)\} \tag{1.1.1}$$

- f(t) … časová funkce
- *F(p)* … operátorový přenos
- L {} … Laplaceova transformace

$$F(p) = \frac{Y(p)}{U(p)} \tag{1.1.2}$$

- U(p) … obraz vstupního signálu
- Y(p) … obraz výstupního signálu

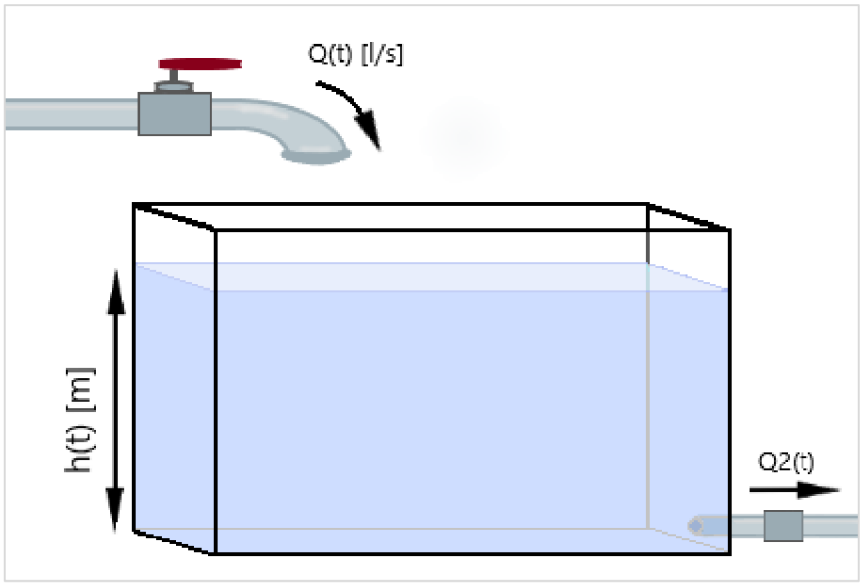
Obecný operátorový přenos standardní formy pro systém prvního řádu:

$$F(p) = \frac{K}{Tp+1} \tag{1.1.3}$$

- K … zesílení systému [-]
- T … časová konstanta [s]

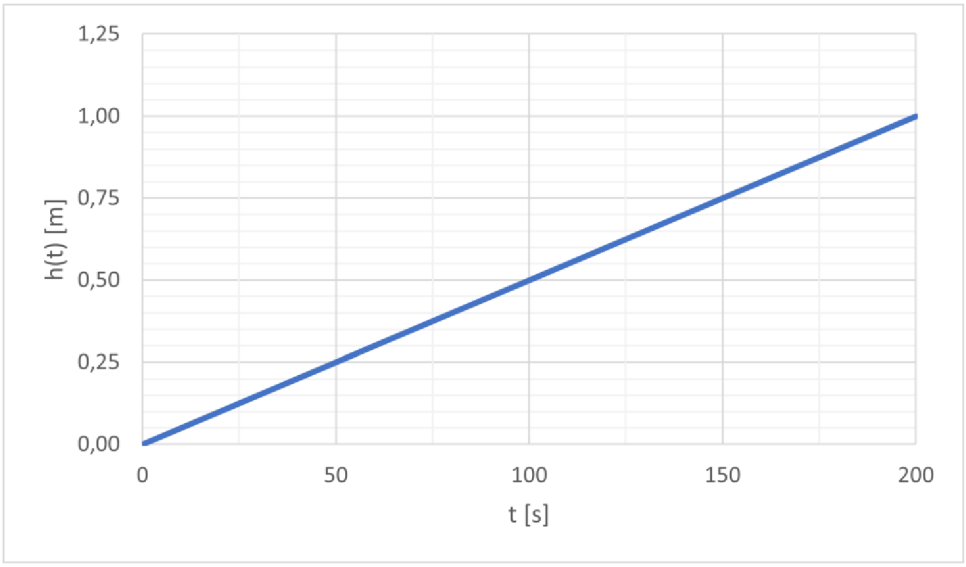
1.1.1 Hydraulický systém

Pro reprezentaci linearity dynamického systému se použila reálná hydraulická soustava. Soustava je tvořena nádobou např. nádrží s výpustí u dna a konstantním uzavíratelným přítokem Obrázek 1.1.1.



Obrázek 1.1.1: Model hydraulické soustavy

V případě, že uvažujeme-li chování systému jako ideální integrátor, nadrž bez výpusti, přivedením jednotkového skoku na vstup systému, tedy otevření kohoutku, nádrž bude napouštěna vodou a výška hladiny lineárně vzroste. Výška hladiny by v ideálních podmínkách rostla do nekonečna Obrázek 1.1.2. Reálný systém omezuje maximální výška nádoby, výška hladiny se ustálí i při stálém průtoku $Q_1(t)$ na hodnotě odpovídající h_{MAX}.



Obrázek 1.1.2: Odezva systému na jednotkový skok

Uvažujeme-li, že součástí soustavy je výpusť umístěná u dna nádrže, popisujeme nelineární systém vyjádřený rovnicí (1.1.4). Nelinearita systému je obsažena v proměnné $Q_2(t)$, kterou vyjadřuje výraz (1.1.5).

$$S_1 h'(t) = Q_1(t) - Q_2(t) \tag{1.1.4}$$

$$Q_2(t) = S_2\sqrt{2gh(t)} \tag{1.1.5}$$

- $Q_1(t)$ … průtok vody z kohoutku [*l/s*]
- $Q_2(t)$ … průtok výpustí [*l/s*]
- S_1 … plocha nádrže [m^2]
- S_2 … plocha výpusti [m^2]
- h(t) … výška hladiny [*m*]
- g … tíhové zrychlení [m/s^2]

Systém linearizujeme dvěma způsoby a dostáváme dvě linearizované diferenciální rovnice. První diferenciální rovnice (1.1.6) parametrizuje nelineární systém tak, že nahradí součin pod odmocninou parametrem *a*.

$$S_1 h'(t) = Q_1(t) - S_2 a h(t) \tag{1.1.6}$$

- *a* … parametrická náhrada

Druhý způsob linearizace systému vyjadřuje diferenciální rovnice (1.1.7), kde průtok $Q_2(t)$ je roven podílu výšky hladiny a hydraulického odporu. Hydraulický odpor považujeme za konstantu nezávislou na proměnné h(t) [2]. Při uvažování tohoto lineárního systému, soustavu doplníme o nastavitelný výpustní ventil, který využíváme v GUI aplikaci.

$$S_1 h'(t) = Q_1(t) - \frac{h(t)}{R} \tag{1.1.7}$$

- R … hydraulický odpor [s/m^2]

Diferenciální rovnici (1.1.7) zapíšeme ve tvaru, který lze implementovat do prostředí MATLAB Simulink a aplikujeme LT, dle popisu převodu na operátorový přenos z kapitoly 1.1. Z operátorového přenosu hydraulického systému určíme časovou konstantu T[s].

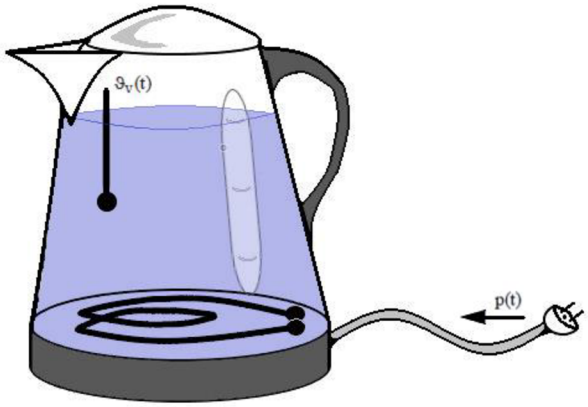
$$S_1 R h'(t) + h(t) = R Q_1(t) \tag{1.1.8}$$

$$F(p) = \frac{Y(p)}{U(p)} = \frac{R}{SRp+1} \tag{1.1.9}$$

$$T = SR\ [s] \tag{1.1.10}$$

1.1.2 Tepelný systém

Reálný tepelný systém prvního řádu použijeme rychlovarnou konvici s topnou spirálou zabudovanou ve dnu. V této soustavě je cílem zjistit změnu a výslednou teplotu ohřívané kapaliny. Tepelná soustava je znázorněna na Obrázku 1.1.3.



Obrázek 1.1.3: Tepelný systém 1. řádu

Diferenciální rovnici zapíšeme ve tvaru (1.1.11), jestliže uvažujeme, že teplota okolí je konstantní a zanedbáváme ji jako časově proměnnou [2]. Též zanedbáváme dynamický ohřev topné spirály.

$$m_v c_v \vartheta'(t) + k_v \vartheta_0 = p(t) \tag{1.1.11}$$

- m_v … hmotnost kapaliny [kg]
- c_v … specifické teplo kapaliny [$Ws/(kg°C)$]
- k_v … koeficient přestupu tepla [$W/°C$]
- p(t) … příkon systému [W]
- $\vartheta(t)$ … teplota kapaliny [$°C$]
- $\vartheta_0(t)$ … počáteční teplota kapaliny [$°C$]

Časovou konstantu tepelného systému odvodíme obdobným postupem, který je ukázán na hydraulickém systému v předešlé podkapitole 1.1.1.

$$(m_v c_v p + k_v) Y(p) = U(p) \tag{1.1.12}$$

$$F(p) = \frac{1}{m_v c_v p + k_v} \tag{1.1.13}$$

Operátorový přenos (1.1.13) není zapsán standartní formou, tedy jmenovatel upravíme tak, aby byl obecný zápis ve tvaru Tp+1. Úpravou přenosu na standartní

formu (1.1.14) ve jmenovateli zjistíme, že časová konstanta je rovna součinu hmotnosti a specifického tepla vody ku koeficientu přenosu tepla (1.1.15).

$$F(p) = \frac{\frac{1}{k_v}}{\frac{m_v c_v}{k_v} p + 1} \tag{1.1.14}$$

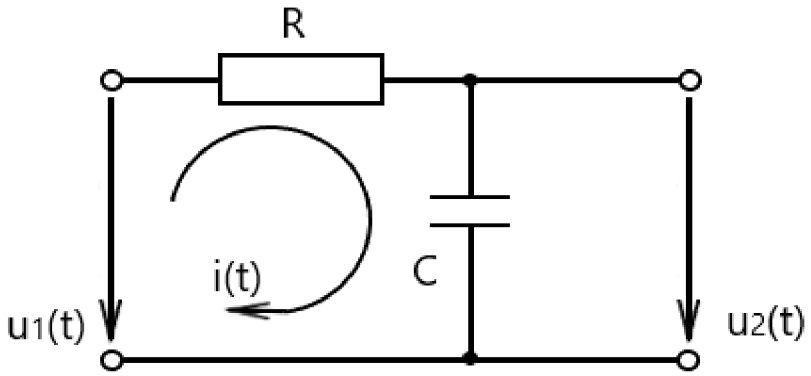
$$T = \frac{m_v c_v}{k_v} \; [s] \tag{1.1.15}$$

Pro implementaci tepelného systému do prostředí MATLAB Simulink upravíme diferenciální rovnici (1.1.11) do tvaru:

$$\vartheta'(t) = \frac{p(t) - k_v \vartheta(t)}{m_v c_v} \tag{1.1.16}$$

1.1.3 Elektrický systém 1. řádu

Elektrický systém prvního řádu reprezentuje RC článek neboli dolní propust. Schéma zapojení RC článku obsahuje rezistor, kondenzátor, vstupní časově proměnné napětí a časově proměnný proud procházející hlavní větví obvodu, který je stejný na rezistoru i kondenzátoru, protože obvod je ekvivalentní zapojení nezatíženého děliče, Obrázek 1.1.4.



Obrázek 1.1.4: Schéma zapojení RC článku

Z jednoduchého elektrického obvodu RC článku dostáváme diferenciální rovnici ve tvaru (1.1.17) [2].

$$RCu_2'(t) + u_2(t) = u_1(t) \tag{1.1.17}$$

- R … odpor [Ω]
- C … kapacita [F]
- $u_2(t)$ … výstupní napětí [V]
- $u_1(t)$ … vstupní napětí [V]

Časovou konstantu z diferenciální rovnice elektrického systému 1. řádu odvodíme obdobným postupem, který je aplikován na tepleném systému v kapitole 1.1.2.

$$F(p) = \frac{Y(p)}{U(p)} = \frac{1}{RCp+1} \tag{1.1.16}$$

$$T = RC\ [s] \tag{1. 1.17}$$

Pro aplikaci do prostředí MATLAB Simulink, upravíme diferenciální rovnici do tvaru, který následně lze implementovat:

$$u_2'(t) = \frac{u_1(t)-u_2(t)}{RC} \tag{1.1.18}$$

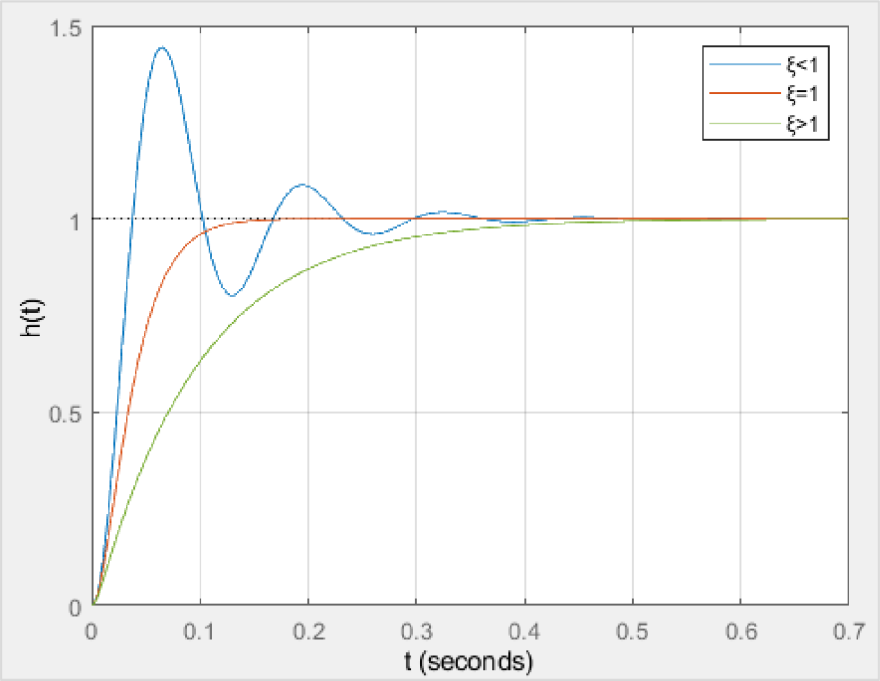
1.2 Systémy 2. řádu

Systémy druhého řádu jsou obecně kmitavými systémy. Obecný operátorový přenos pro systémy 2. řádu definujeme podle výrazu (1.2.1).

$$F(p) = \frac{K}{T_1^2 p^2+2\xi T_1 p+1} \tag{1.2.1}$$

- T_1 … vlastní perioda kmitů
- ξ … koeficient poměrného tlumení
- K … zesílení

Přechodová charakteristika zobrazuje odezvu systému, pokud na vstup přivedeme jednotkový skok. Využitím přechodové charakteristiky zobrazíme všechny tři typy systémů druhého řádu viz Obrázek 1.2.1



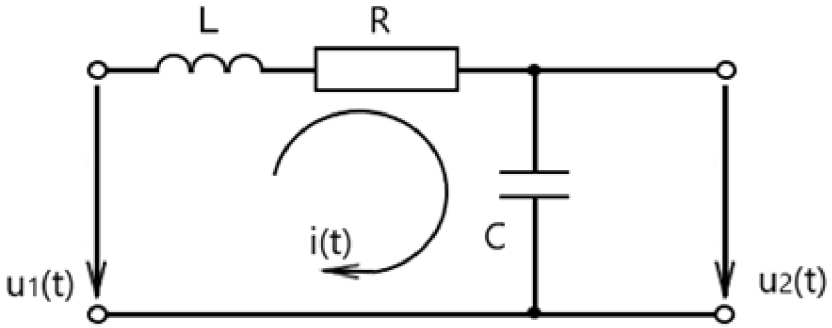
Obrázek 1.2.1: Přechodová charakteristika systému 2. řádu

Kmitavé systémy dělíme do tří skupin podle hodnoty ξ [3]:

- **Kmitavý** – pokud hodnota ξ leží v rozsahu 0 až 1. Systém bude tlumeně kmitat do doby, než se ustálí na zesílení systému K.
- **Aperiodický** – vzniká v případě, že vypočtená hodnota ξ je větší než jedna.
- **Mez aperiodický** – je-li hodnota ξ rovna 1, systém je na mezi aperiodicity.

1.2.1 Elektrický systém 2. řádu

RLC odvod zastupuje elektrický systém 2. řádu. Systém tvoří tři elektronické součástky rezistor, kondenzátor a cívka. Schéma zapojení na Obrázek 1.2.2. Do systému též vstupuje napětí, které napájí celý obvod. Pro RLC obvod platí stejné podmínky jako pro RC článek v kapitole 1.1.3. Diferenciální rovnici elektrického systému zapíšeme ve tvaru (1.2.2) [2].



Obrázek 1.2.2: Schéma zapojené RLC obvodu

$$LCu_2''(t) + RCu_2'(t) + u_2(t) = u_1(t) \tag{1.2.2}$$

- L … indukčnost [H]
- C … kapacita [F]
- R … odpor [Ω]
- $u_2(t)$ … výstupní napětí [V]
- $u_1(t)$ … vstupní napětí [V]

Koeficient poměrného tlumení určíme převedením diferenciální rovnice elektrického systému 2. řádu na operátorový přenos systému druhého řádu.

$$F(p) = \frac{Y(p)}{U(p)} = \frac{1}{LCp^2 + RCp + 1} \tag{1.2.3}$$

Ze zápisu není hned jasné, jaké hodnoty koeficient poměrného tlumení ξ nabývá. Musíme jmenovatel upravit tak, aby vyjadřoval ξ. Porovnáním obecného operátorového přenosu a přenosu RLC obvodu dostaneme:

$$T^2 = LC \rightarrow T = \sqrt{LC} \tag{1.2.4}$$

$$2\xi T = RC \rightarrow \xi = \frac{RC}{2T} \tag{1.2.5}$$

Z odvozených rovnic (1.2.4) a (1.2.5) dostaneme po dosazení obecný zápis koeficientu poměrného tlumení pro RLC obvod ve tvaru:

$$\xi = \frac{RC}{2\sqrt{LC}} \tag{1.2.6}$$

Pro aplikaci do prostředí MATLAB Simulink, upravíme diferenciální rovnici do tvaru, který následně lze implementovat:

$$u_2''(t) = \frac{1}{LC}\left[u_1(t) - \left(RCu_2'(t) + u_2(t)\right)\right] \tag{1.2.7}$$

2. IMPLEMENTACE MODELŮ VYBRANÝCH SYSTÉMŮ V PROSTŘEDÍ MATLAB SIMULINK

K implementaci navržených systémů budeme používat aplikaci MATLAB Simulink. Tato aplikace využívá jednoduchého blokového zápisu, vyjadřující diferenciální rovnice, které jsme již odvodili v předchozích podkapitolách 1.1 a 1.2.

Začít s implementací do aplikace MATLAB Simulink můžeme, když známe hodnoty všech proměnných v diferenciálních rovnicích. V prostředí využijeme jen základní bloky pro realizaci rovnic. Těmito bloky jsou Step, Gain, Sum, Integrator a Scope.

2.1 Implementace a ověření systému 1. řádu

V této kapitole implementujeme do prostředí Simulink hydraulický, tepelný a elektrický systém prvního řádu. Na konci kapitoly porovnáme všechny tři systémy pomocí jejich časových konstant, které jsme si předem odvodili v podkapitolách 1.1.1, 1.1.2 a 1.1.3. Pro implementaci využijeme již odvozených diferenciálních rovnic v kapitole 1.1.

2.1.1 Implementace hydraulického systému

V kapitole 1.1 odvozujeme tři diferenciální rovnice hydraulického systému. Pro realizaci systému do prostředí Simulink linearizujeme reálně nelineární hydraulickou soustavu, kde přepokládáme tyto případy.

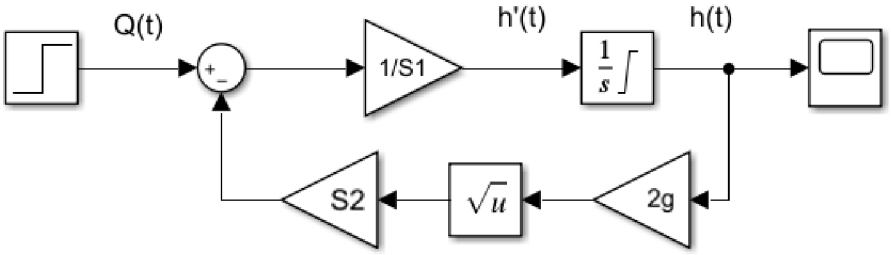
- Nádrž bez regulace výpustním ventilem
- Nádrž s nastavitelným výpustním ventilem

Reálné parametry systému budou v obou případech stejné. Z diferenciálních rovnic víme, že systém využívá tyto parametry o zvolených hodnotách:

- Rozměry nádrže DxŠxV 1x0.6x0.6 m
- Plocha nádrže $S_1=0.6\ m^2$
- Průtok $Q_1(t)=0.8\ l/s = 0.0008\ m^3/s$
- Hydraulický odpor $R \in (1;5000)\ s/m^2$
- Tíhové zrychlení $g=9.81\ m/s^2$
- Plocha výpusti $S_2=3.14\ cm^2$
- Parametr *a* $a \in (1;100)$

Nádrž s výpustí implementujeme do prostředí z výše stanovenými parametry a dle odvozených diferenciálních rovnic (1.1.3) a (1.1.4). Nelineární model soustavy uvažuje neregulovatelnou výpust, jelikož zpětnou vazbu tvoří konstanty, jak znázorňuje schéma zapojení systému v prostředí MATLAB viz Obrázek 2.1.1.

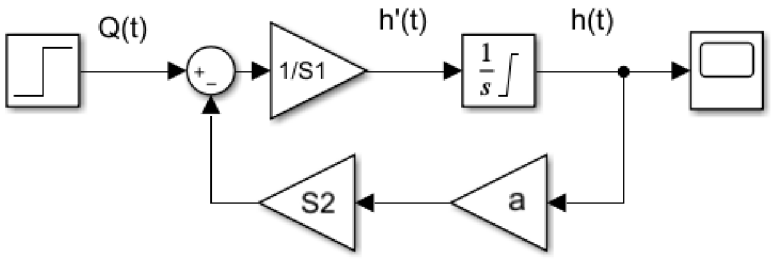
V modelu dále uvažujeme, že plocha odtoku má tvar kruhu o průměru d=2 cm. Jelikož systému obsahuje odmocninu, v schématu musí být zajištěno, aby odmocňovaný výraz nenabýval záporné hodnoty. Toho lze docílit přidáním bloku, který vytvoří absolutní hodnotu výrazu před jeho odmocněním.



Obrázek 2.1.1: Schéma nelineárního systému MATLAB Simulink

Parametrizováním předcházejícího modelu linearizujeme hydraulický systém, odstraněním přirozené nelinearity, odmocniny. Parametr *a* je určen rozsahem hodnot, protože nelze přesně určit hodnotu odpovídající výrazu (2.1.1), jelikož obsahuje s časem proměnou výšku hladiny. Hlavním důvodem je využití zadaného rozsahu k tvorbě aplikace GUI. Následně parametru *a* odpovídá jednotka [s^{-1}]. Dalo by se tedy říct, že parametr představuje frekvenci. Tato skutečnost vyplývá z nutnosti zachování jednotek vstupujících do sumačního bloku.

$$\sqrt{2gh(t)} \tag{2.1.1}$$

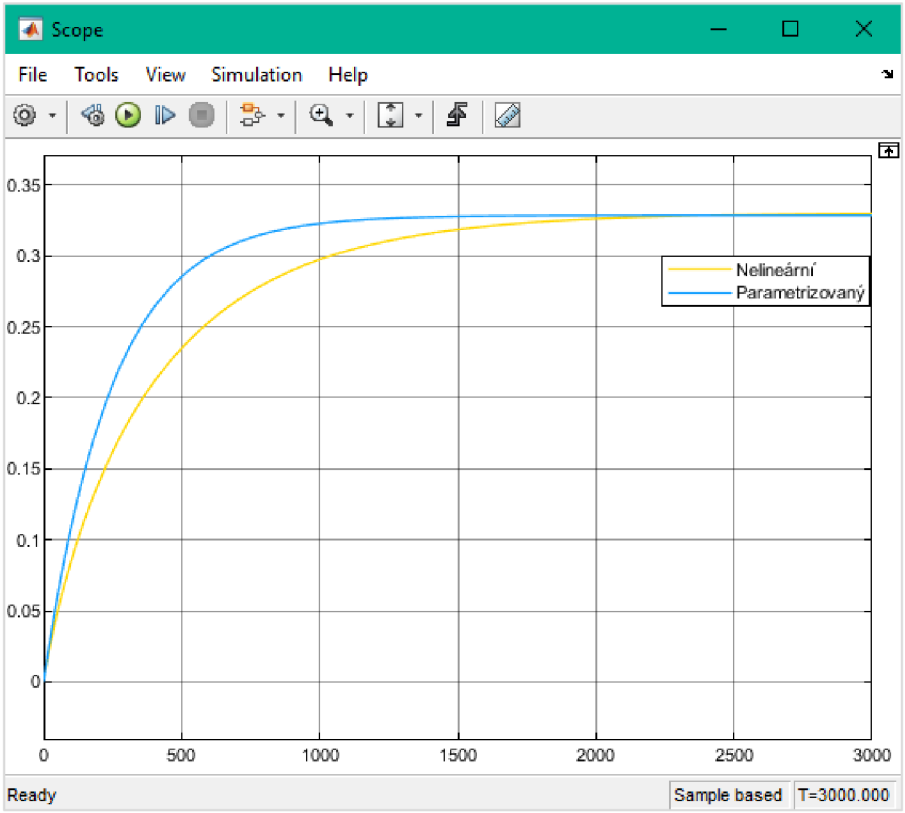


Obrázek 2.1.2: Schéma parametrizovaného systému MATLAB Simulink

Když jsou systémy sestrojeny, spustíme simulaci a otevřeme výstupní závislosti blokem Scope viz Obrázek 2.1.3. Graf závislosti nelineárního systému ukazuje, že výška hladiny se ustálí na hodnotě h(t)=0.33 m, tedy přibližně v polovině celkové výšky nádrže při stanovených parametrech.

K linearizovanému systému musí být provedena série simulací pro učení správné hodnoty parametru *a*, který by odpovídal co nejpřesněji nelineárnímu modelu. První simulace byla spuštěna s hodnotou *a* rovnou 4.43 s^{-1}. Tuto hodnotu určíme vypočtením výrazu (2.1.1), kde do výpočtu neuvažujeme proměnou h(t). Výsledný graf se ustálí na hodnotě 0.57 m a simulovanou nádrž zcela zaplnila voda. V následujících simulacích se hodnota parametru zvyšovala o 0.5, to způsobilo pokles výšky hladiny h(t). Při nabytí

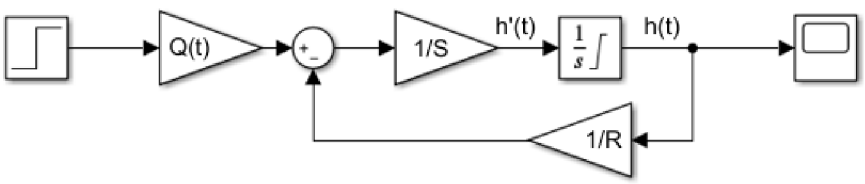
hodnoty parametru a = 8 s^{-1}, výsledný graf nabyl hodnoty pod úrovní nelineárního systému, proto nastalo snižování hodnoty a zjemnění kroku na 0.05, dále jen na krok 0.025. Hodnotu parametru jsme stanovili na a = 7.725 s^{-1}.



Obrázek 2.1.3: Závislost výšky hladiny na čase nelineárního a parametrizovaného systému

Nádrž s nastavitelným výpustním ventile implementujeme do prostředí MATLAB Simulink z výše stanovenými parametry a dle odvozené diferenciální rovnice (1.1.8). V modelu uvažujeme, že součásti výpusti je výpustní ventil s průtokem $Q_2(t)$. Průtok výpustního ventilu lze vyjádřit rovnicí (2.1.2), kde vztahujeme výšku hladiny h(t) vůči hydraulickému odporu. Simulujeme tedy systém, ve kterém na 0.6 m výšky odteče úměrné množství vody v závislosti na proměnné R.

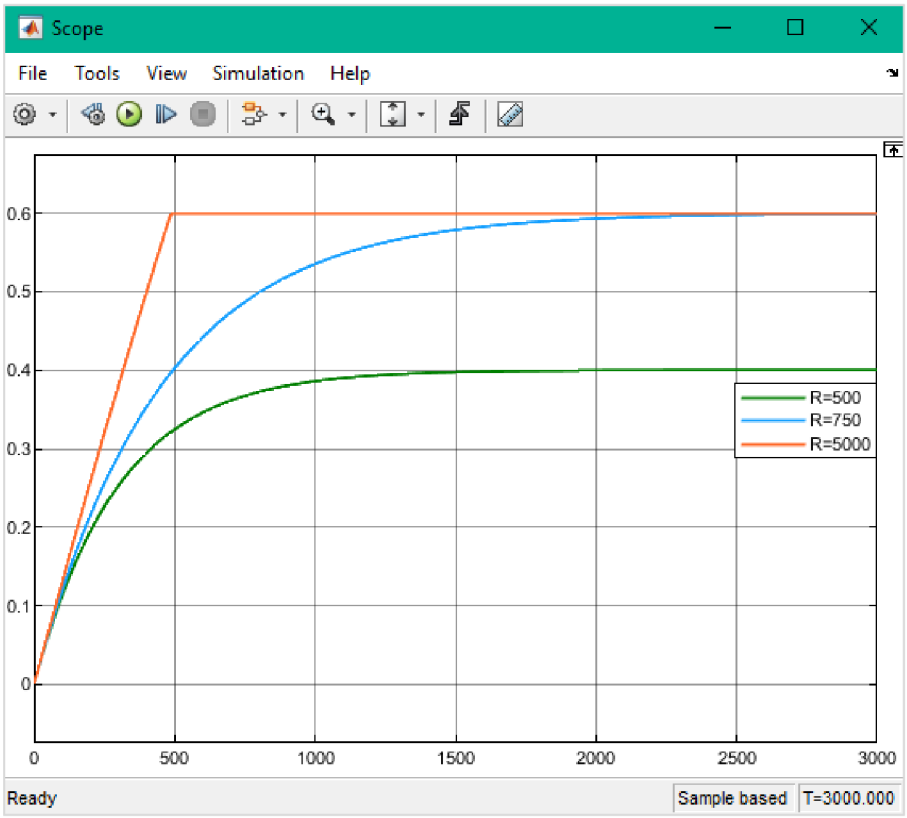
$$Q_2(t) = \frac{h(t)}{R} \tag{2.1.2}$$



Obrázek 2.1.4: Schéma linearizovaného hydraulického systému

V simulaci můžou nastat 3 možnosti výstupu ze simulace (Obrázek 2.1.5):

- **Průtok $Q_2(t) = 0\ m^3/s$** – je čistě lineární a chová se jako integrátor, který je omezený jen výškou nádrže. Proměnná R musí nabývat vysoké hodnoty. V simulaci jsme zvolili hodnotu R=5000 s/m^2.
- V této simulaci se systém snažíme vyvážit tak, aby při ustálení dosáhl mezní hodnoty. K vyrovnání vstupního a výpustního průtoků nastane těsně před hranou nádrže bez toho, aby nádrž přetekla. Vypočtený hydraulický odpor při h(t)=0,6 m, nabývá hodnoty R=750 s/m^2.
- Výsledkem poslední simulace může být více přechodových charakteristik. Systém zde může nabývat v ustáleném stavu výšky hladiny v rozmezí $h(t) \in (0;0.6)$ m. V simulaci uvažujeme, že nádrž naplníme do dvou třetiny maximální výšky. Vypočtený hydraulický odpor pro h(t)=0.4 m odpovídá hodnotě R=500 s/m^2.



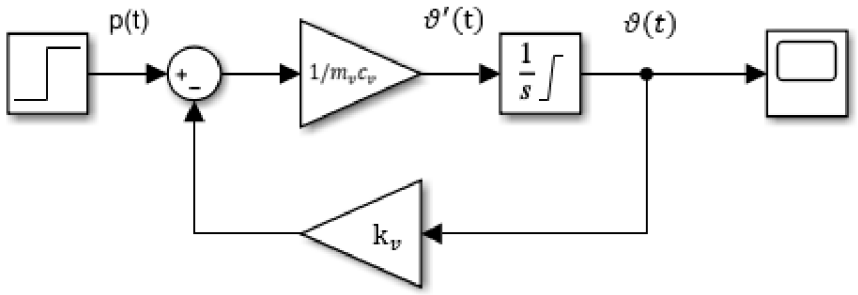
Obrázek 2.1.5: Závislosti výšky hladina na čase linearizovaného hydraulického systému

2.1.2 Implementace tepelného systému

Pro tvorbu tepelného systému v prostředí MATLAB Simulink potřebujeme nejdříve určit parametry, se kterými simulace zpracuje výstupní závislost teploty vody v rychlovarné konvici na čase. Celý systém vychází z diferenciální rovnice (1.2.9).

Parametry tepelného systému:

- Hmotnost kapaliny $m_v = 1.7\ l = 1.7\ kg$
- Specifické teplo kapaliny $c_v = 4180 Ws/(kg°C)$ [4]
- Koeficient přestupu tepla $k_v = 0.8\ W/°C$ [4]
- Příkon systému $p(t) = 2200\ W$
- Teplota kapaliny $\vartheta_{MAX} = 100\ °C$
- Počáteční teplota kapaliny $\vartheta_0(t) = 10\ °C$

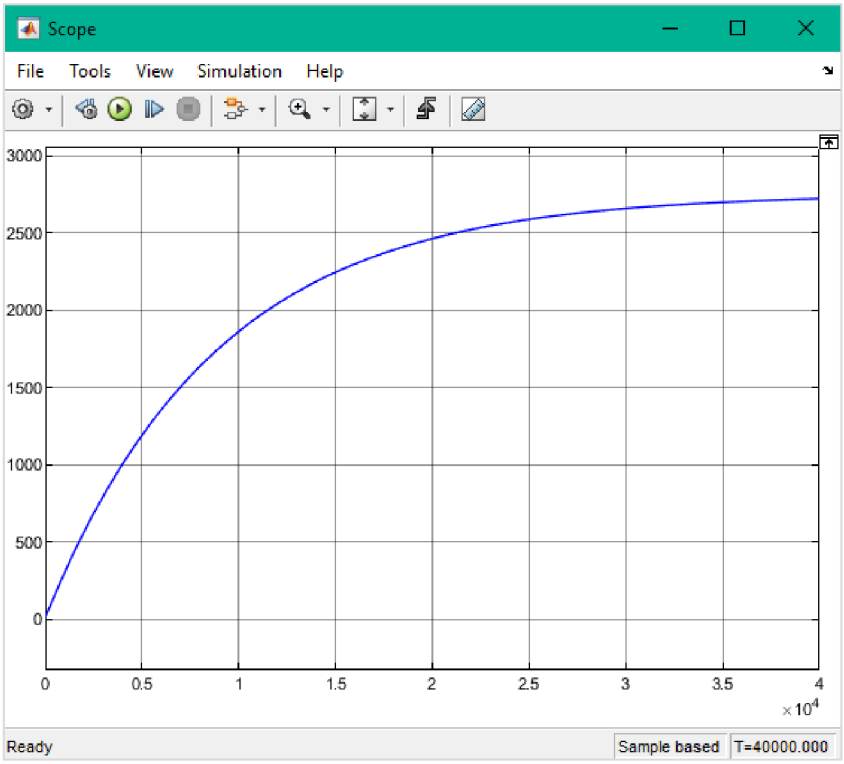


Obrázek 2.1.6: Schéma linearizovaného tepelného systému

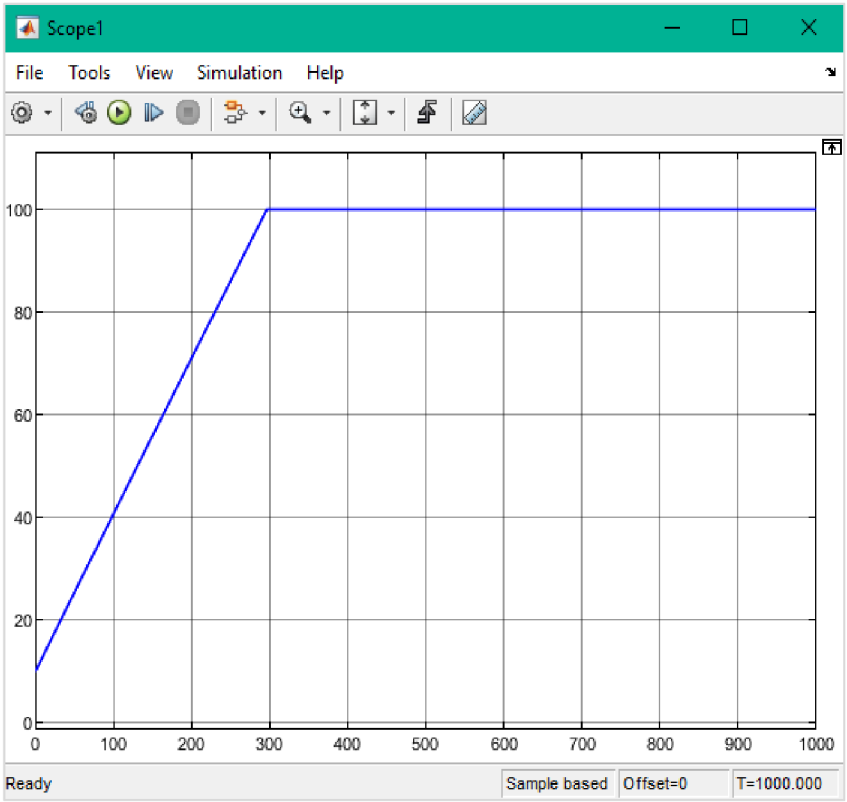
Výsledný graf tepelného systému tvarem odpovídá systému prvního řádu, ale výsledné hodnoty teploty jsou mnohonásobně vyšší, řádově o tisíce viz Obrázek 2.1.8. Důvodem tohoto chování modelu je, že simulovaný systém neuvažuje ve výpočtech základní fyzikální vlastnosti kapalin, například vypařování.

Abychom realizovali reálné chování systému musíme do modelu přidat omezení, které nasimuluje bod varu, kde dochází k přeměně skupenství z kapalného na plynné. Změnu provedeme zjednodušeně například nastavením saturace v bloku Integrátor, povolením limit a nahrazením předdefinované hodnoty Inf hodnotou 100. Výslednou závislost lze vidět na Obrázku 2.1.7.

Z grafu omezeného bodem varu kapaliny lze téže vyčíst čas potřebný pro ohřátí 1.7 l vody, při příkonu rychlovarné konvice 2200 W. Čas ohřevu je roven pěti minutám, což odpovídá reálnému času sepnutí tepelného čidla v rychlovarné konvici.



Obrázek 2.1.8: Graf závislosti teploty kapaliny na čase linearizovaného tepelného systému



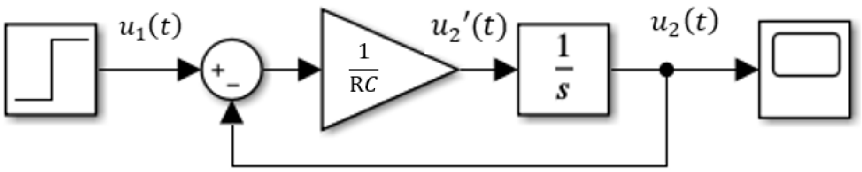
Obrázek 2.1.7: Graf závislosti teploty kapaliny na čase omezený bodem varu

2.1.3 Implementace elektrického systému 1. řádu

V prostředí MATLAB Simulink vytvoříme model RC článku. Následně určíme potřebné parametry elektrických součástek, se kterými simulace zpracuje závislost výstupního napětí na čase. Celý model vychází z diferenciální rovnice (1.1.18).

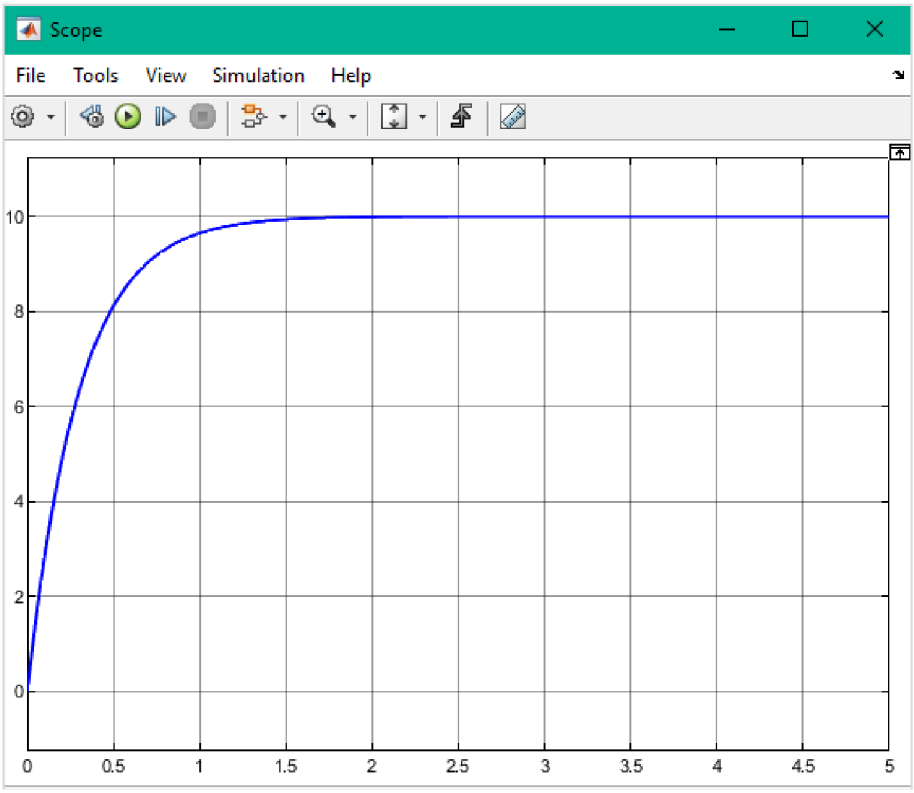
Definované parametry elektrického RC článku:

- Kapacita C= 2 mF
- Elektrický odpor R= 150 Ω
- Vstupní napětí $u_1(t)$= 10 V



Obrázek 2.1.9: Schéma linearizovaného RC článku

Graf závislosti z prostředí Simulink odpovídá předpokládanému výsledku. Na obrázku 2.1.10 lze vidět přechodovou charakteristiku systému prvního řádu. Výstupní napětí nabývá maximální hodnoty $u_2(t)$ = 10 V.

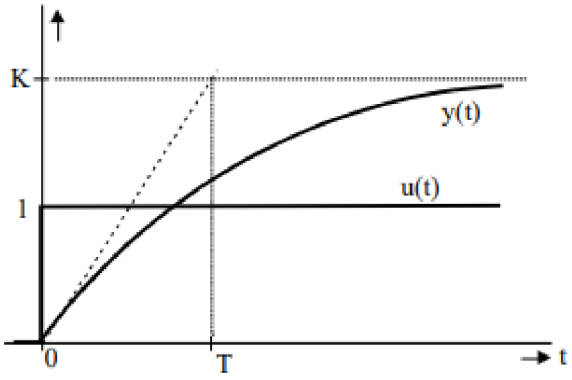


Obrázek 2.1.10: Graf závislosti výstupního napětí na čase RC článku

2.1.4 Ověření časových konstant simulovaných systému 1. řádu

Z výše vykreslených závislostí, lze též zjistit časovou konstantu systému. Můžeme ověřit, zda teoretická hodnota časové konstanty odpovídá reálné v našem případě simulované. Tuto vlastnost využijeme pro kontrolu implementovaných systémů.

Časovou konstantu lze získat dvěma způsoby. Jedna z metod používá tečnu vedenou z počátku vstupního signálu až do bodu, kde tečna protne pomyslnou přímku zesílení K vodorovnou s osou x. Z průsečíku přímek spustíme kolmici. Časová konstanta odpovídá hodnotě na ose x, kde ji protnula kolmice viz Obrázek 2.1.11.



Obrázek 2.1.11: Reakce systému na jednotkový skok [2]

Druhá metoda vychází z předcházející tím, že aproximujeme přechodovou charakteristiku simulovaného systému [5]. Neznámá časová konstanta náleží 63 % ustálené výstupní hodnotě systému. Vynásobíme-li maximální dosaženou hodnotu systému 0.63 (2.2.1), tak příslušná hodnota na ose x odpovídá časové konstantě systému [6].

$$y_x(t) = 0{,}63 \cdot y_{MAX}(t) \tag{2.2.1}$$

- $y_x(t)$… výstupní signál v 63 %
- $y_{MAX}(t)$… maximum výstupního signálu

Pro ověření teoretických časových konstant využíváme druhou metodu. Teoretické časové konstanty a 63% hodnoty výstupních signálů ze systému vypočteme následně:

- Hydraulický systém vychází z výrazu (1.2.11)

$$T_{h(t)V} = SR = 0{,}6 \cdot 750 = 450\ s \tag{2.2.2}$$

$$h_{63\%}(t) = 0{,}63 \cdot h_{MAX}(t) = 0{,}63 \cdot 0.6 = 0{,}378\ m \tag{2.2.3}$$

- $T_{h(t)V}$… teoretická časová konstanta hydraulického systému
- $h_{63\%}(t)$… výšky hladiny v 63 %
- $h_{MAX}(t)$… maximální výška hladiny

- Tepelný systém vychází z výrazu (1.2.8)

$$T_{\vartheta(t)V} = \frac{m_v c_v}{k_v} = \frac{1{,}7 \cdot 4180}{0{,}8} = 8882{,}5\ s \tag{2.2.4}$$

$$\vartheta_{63\%}(t) = 0{,}63 \cdot \vartheta_{MAX}(t) = 0{,}63 \cdot 2758{,}7 = 1738\ ℃ \tag{2.2.5}$$

- $T_{\vartheta(t)V}$… teoretická časová konstanta tepelného systému
- $\vartheta_{63\%}(t)$… teplota vody v 63 %
- $\vartheta_{MAX}(t)$… maximální teplota vody

- Elektrický systém vychází z výrazu (1.2.14)

$$T_{u_2(t)V} = RC = 150 \cdot 0{,}002 = 0{,}3\ s \tag{2.2.6}$$

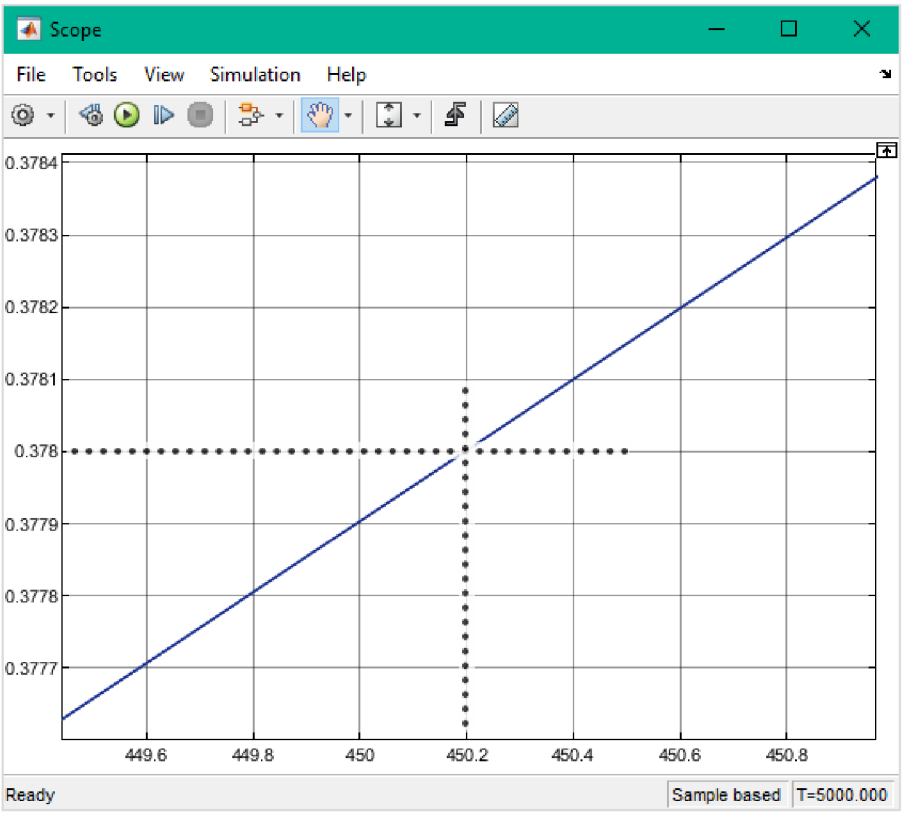
$$u_{2_{63\%}}(t) = 0{,}63 \cdot u_{2MAX}(t) = 0{,}63 \cdot 10 = 6{,}3\ V \tag{2.2.7}$$

- $T_{u2(t)V}$… teoretická časová konstanta elektrického systému
- $u_{263\%}(t)$… výstupní napětí v 63 %
- $u_{2MAX}(t)$… maximální výstupní napětí

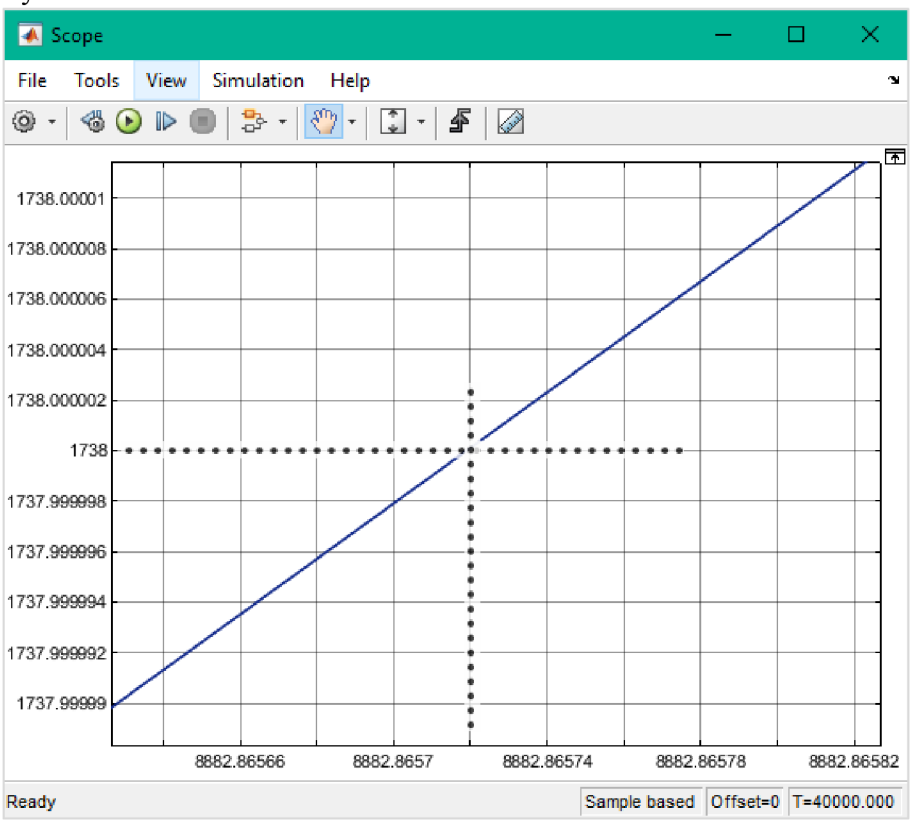
Časové konstanty simulovaných modelů získáme odečtením hodnot z vykreslených přechodových charakteristik. V již otevřeném bloku Scope zvětšíme oblast grafu okolo hledané hodnoty na ose y. V místě přiblížení nalezneme hodnotu odpovídající 63 % signálu a z osy x odečteme příslušné číslo-časovou konstantu modelovaného systému. Grafické odečtení časových konstant reprezentují obrázky – Obrázek 2.1.12, Obrázek 2.1.13 a Obrázek 2.1.14.

Odečtené časové konstanty z přechodových charakteristik:

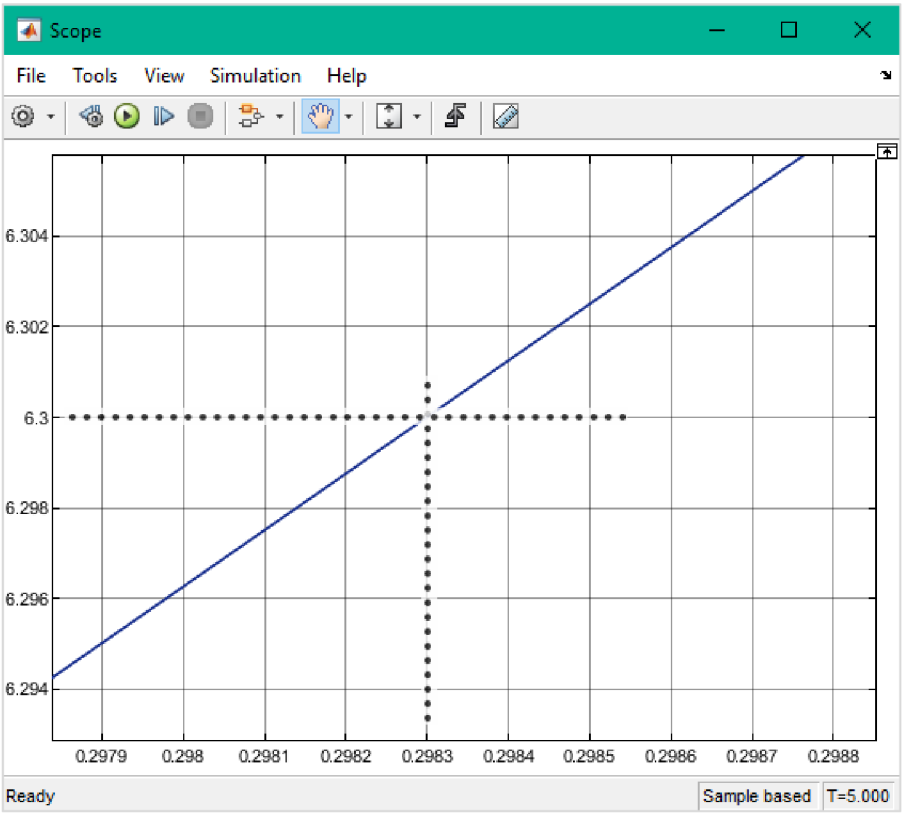
- Hydraulický systém – $T_{h(t)} = 450.2$ s
- Tepelný systém – $T_{\vartheta(t)} = 8882.86$ s
- Elektrický systém – $T_{u2(t)} = 0.2983$ s



Obrázek 2.1.12: Grafické odečtení časové konstanty hydraulického systému



Obrázek 2.1.13: Grafické odečtení časové konstanty tepelného systému



Obrázek 2.1.14: Grafické odečtení časové konstanty elektrického systému

Porovnáme-li teoretické hodnoty časových konstant s odečtenými z přechodových charakteristik jednotlivých systémů, zjistíme že odchylka mezi nimi není větší než desetiny vteřiny. Simulovaný hydraulický systém v porovnání s teoretickou časovou konstantou vykazuje rozdíl 0.2 s. Tepelný systém oproti ostatních systém má největší rozdíl mezi časovými konstantami a to 0.36 s. Poslední elektrický systém naopak vykazuje nejmenší odchylku mezi konstantami. Rozdíl činí 0.0017 s, mohli bychom tedy říct, že rozdíl mezi časovými konstantami je zanedbatelný.

Odchylky mezi časovými konstantami můžou být zapříčiněny například metodou, zaokrouhlováním nebo nepřesným odečtem z přechodových charakteristik.

2.2 Implementace systému 2. řádu

Pro tvorbu elektrického systému druhého řádu v prostředí Simulink potřebujeme nejdříve určit parametry, které zastoupí všechny tři typy přechodových charakteristik.

Hodnoty elektrických součástek jednotlivých typů určíme výpočtem koeficientu poměrného tlumení (1.2.6). Nejprve vypočteme hodnoty pro aperiodický a kmitavý systém. Následně vyvážíme hodnoty součástek pro systém na mezi aperiodicity. Simulovaný model vychází z diferenciální rovnice (1.2.7).

$$\xi = \frac{RC}{2\sqrt{LC}} \tag{1.2.6}$$

Parametry elektrického RLC systému:

- **ξ >1** – Aperiodický systém

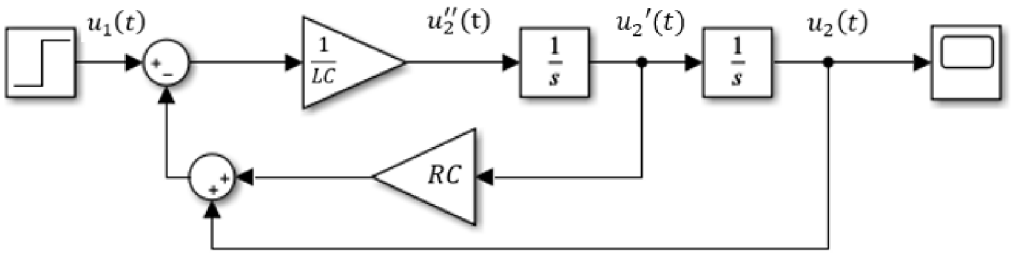
- Kapacita C= 2 mF
- Elektrický odpor R= 50 Ω
- Indukčnost L= 0.2 H
- Koeficient poměrného tlumení ξ = 2.5
- Vstupní napětí $u_1(t)$= 10 V

- **ξ = 1** – Systém na mezi aperiodicity

- Kapacita C= 2 mF
- Elektrický odpor R= 20 Ω
- Indukčnost L= 0.2 H
- Vstupní napětí $u_1(t)$= 10 V

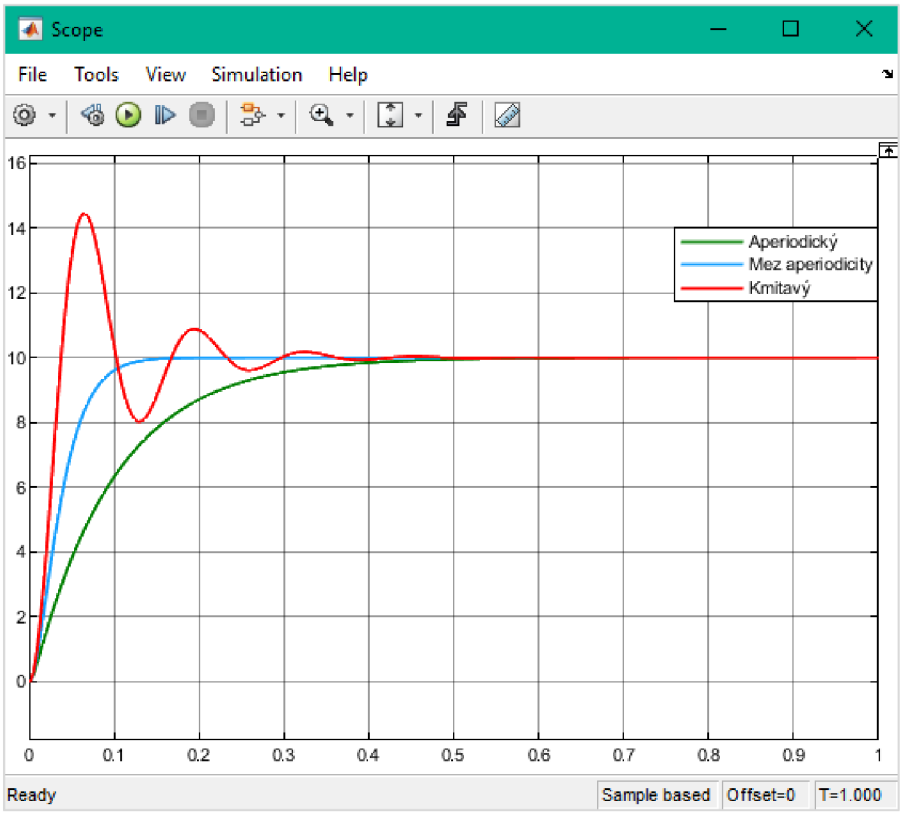
- **ξ ϵ (0;1)** – Kmitavý systém

- Kapacita C= 2 mF
- Elektrický odpor R= 5 Ω
- Indukčnost L= 0.2 H
- Koeficient poměrného tlumení ξ = 0.25
- Vstupní napětí $u_1(t)$= 10 V



Obrázek 2.2.1: Schéma linearizovaného elektrického systému 2. řádu

Na obrázku 2.2.2 můžeme vidět všechny tři stavy systému, kterých systémy 2. řádu můžou nabývat. Přechodová charakteristika aperiodického systému s koeficientem poměrného tlumení ξ = 2.5, dle předpokladů potvrzuje, že systém je přetlumený. Aperiodický systém dosáhne maximální hodnoty později než vyvážený systém na mezi aperiodicity. Naopak přechodová charakteristika kmitavého systému s koeficientem poměrného tlumení ξ = 0.25, osciluje okolo maximální hodnoty výstupního napětí s postupně zmenšujícím se překmitem. Dle předpokladu systém nemá dostatečné tlumení, a proto dochází k rozkmitání výstupního signálu.



Obrázek 2.2.2: Graf závislosti výstupního napětí na čase RLC systému

3. GUI V PROSTŘEDÍ MATLAB

V prostředí MATLAB 2020b využijeme aplikaci pro tvorbu GUI – grafické uživatelské rozhraní. Aplikace umožňuje vytvářet 2D nebo 3D obraz systému a následně provádět simulaci v závislosti na napsaném programu. Výhodou GUI může být, že při užívání aplikace stačí jen základní znalosti programovacího jazyka.

Aplikaci GUI lze vytvořit třemi způsoby, podle potřeby vývojáře [7].

- **Programově tvořená aplikace** – tento způsob se využívá pro větší kontrolu nad vytvářenou aplikací. Aplikace se píše za pomoci funkcí MATLABu. Po vytvoření nového skriptu vytvoříte funkci a následně zde programujeme komponenty a chování celé aplikace.
- **Life skript** – možnost živého skriptu zvolíme v případě, kdy máme vytvořený skript a chceme umožnit uživatelům snadný přístup k proměnným v kódu.
- **App Designer** – tato metoda vytváření GUI umožní interaktivně vytvářet aplikace. App Designer je prostředí v MATLABu podobně jako prostředí Simulink. Zde pracujeme se dvěma rozhraními, v jednom utváříme vzhled za pomocí komponent a v druhém utváříme chování celé aplikace.

3.1 Tvorba App Designer

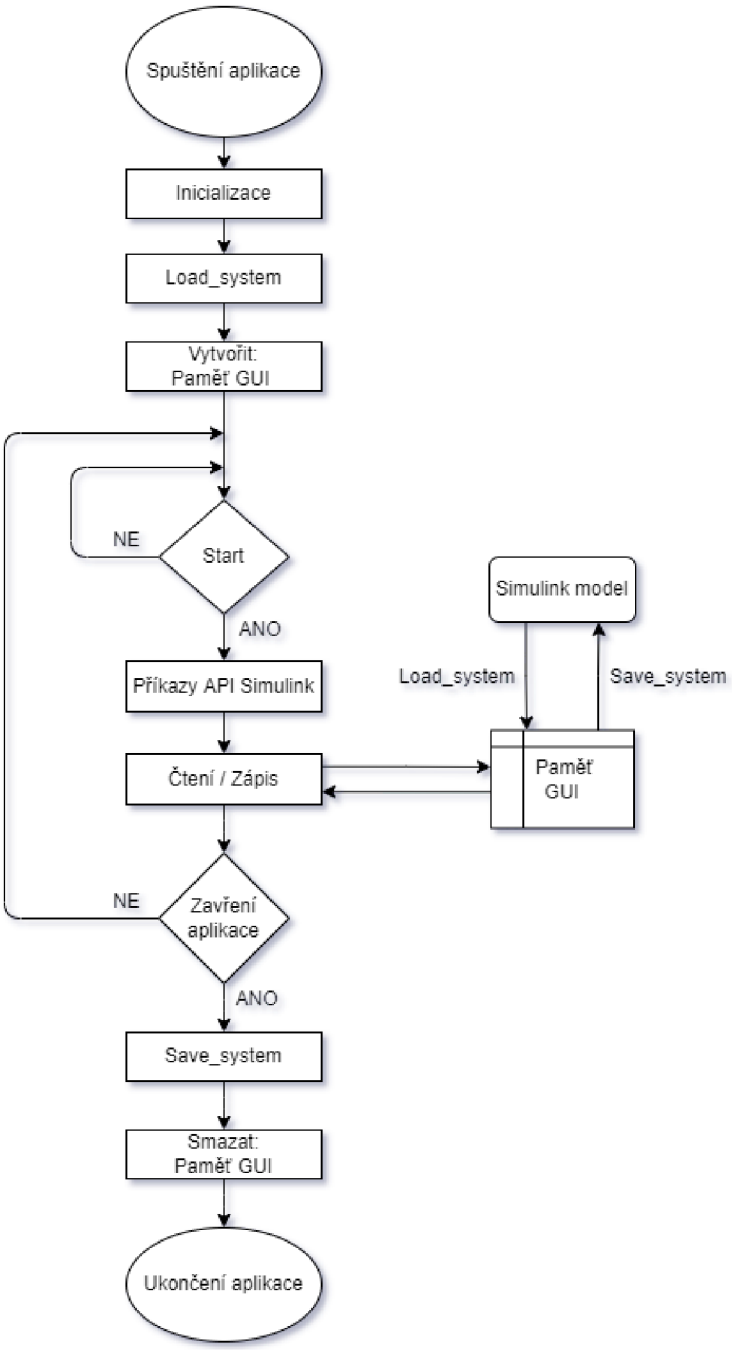
V nejnovější verzi MATLABu nevoláme aplikaci GUI pomocí Command Window, ale přistupujeme přes záložku v horní liště s názvem APPS a vybráním ikony Design App.

Podobně jako v prostředí Simulink se vytvoří nové okno pro založení nebo otevření rozpracovaného projektu. Prostředí nabízí tři typy rozvržení při zakládání nového projektu, a to Blank App, 2-Panel App with Auto-Reflow nebo 3-Panel App with Auto-Reflow.

Prostředí App Designer je rozvrženo do několika sekcí. Na levé straně rozhraní nalezneme sekci s knihovnou komponent – Component library, na pravé straně pak sekci s přehledem využitých prvků a jejich nastavením – Component browser. Uprostřed nalezneme pracovní plochu, do které vkládáme prvky z knihovny komponent, a vytváříme tím vzhled celé aplikace. Do sekce pro psaní kódu se dostaneme přepnutím tlačítka v pravém horním rohu nad pracovní plochou, následkem přepnutí se na levé straně změní sekce z knihovny na sekci s přehledem použitých funkcí a callbacků v kódu.

3.2 Propojení prostředí Simulink a App Designer

K propojení aplikace App Designer a modelu Simulink slouží příkazy aplikačního programovacího rozhraní Simulink, dále jen API Simulink (Application Programming Interface). Příkazů API Simulink, které lze využít k propojení nebo ovládání Simulink modelu přes vytvořenou aplikaci App Designeru, je velké množství s různými možnostmi využití [8].



Obrázek 3.2.1: Principiální propojení aplikace App Designer a modelu Simulink

Na obrázku 3.2.1 vidíme diagram principiálního propojení aplikace App Designeru a modelu Simulink. Při spuštění aplikace, nejprve proběhne inicializace všech objektů v aplikaci. Následně se z kódu aplikace načte příkaz load_system, který vytvoří novou paměť a uloží do ní příslušný systém [9]. Obdobně by šlo využít i příkaz open_system, s tím rozdílem, že při vykonávání tohoto příkazu otevře požadovaný Simulink model v novém okně [10]. Oba příkazy jsou podmínkou pro přístup do modelu ostatními příkazy API Simulink.

Když je aplikace načtená, čeká se na splnění podmínky v kódu, která povolí vykonání další části kódu, a tedy i použití jiných příkazů API Simulink, například těmito příkazy může být set_param, get_param nebo příkaz sim. Na základě charakteru příkazu se do paměti s načteným systémem zapisuje nebo čte žádaná hodnota proměnné nebo výstup celého systému.

Propojení mezi App Designerem a Simulink modelem potrvá, dokud nedojde k zavření aplikace. V ten okamžik se v kódu zavolá příkaz save_system a načtený model v paměti se uloží, poté se paměť vymaže [11]. Příkaz save_system použijeme jak při využití příkazu load_system, tak u příkazu open_system. K uložení načteného modelu lze též použít příkaz close_system, vhodné je ho využívat, pokud jsme použili pro načtení prostředí příkaz open_system, jelikož uzavře i otevřené okno systému, které se otevřelo při spuštění aplikace [12].

Tohoto principu propojení využíváme při tvorbě aplikací v kapitole 5, kde přistupujeme k výše vytvořeným modelům v prostředí Simulink.

4. NÁVRH APLIKACÍ PRO PROSTŘEDÍ APP DESIGNER

V předchozích kapitolách jsme vybrali reálné systémy s různými fyzikálními vlastnostmi a implementovali je to prostředí MATLAB Simulink. Následně jsme provedli simulace jednotlivých systémů.

V této kapitole navrhneme aplikace, které umožní ovládat modely v prostředí Simulink, měnit parametry a vykreslit přechodové charakteristiky simulovaných systémů bez nutnosti modely otevřít.

Simulované systémy v prostředí MATLAB Simulink rozdělíme do čtyř GUI aplikací úloh, které obsahují:

- **Úloha 1** – Porovnání hydraulického nelineárního systému s linearizovaným systém s parametrickou náhradou
    - Aplikace slouží k poukázání, jak dva způsoby vyjádření stejného hydraulického systému viz kapitola 1.1.1 a 2.1.1, ovlivní výstup z implementovaných modelů. Zároveň se snaží uživateli přiblížit rozdíl mezi nelineárním systémem a linearizovaným s parametrickou náhradou, při snaze určit hodnotu parametrické náhrady tak, aby linearizovaný systém co nejvíce odpovídal nelineárnímu systému.

- **Úloha 2** – Linearizovaný hydraulický systém s výpustním ventilem
    - Účel aplikace je ukázat, jak výpustní ventil nádrže dokáže ovlivnit systém při jeho souběžném napouštění. Současně ukazuje rozdíl v rychlosti plnění nádrže při zavřené a otevřené výpusti i v závislosti na velikosti nádrže.

- **Úloha 3** – Linearizovaný tepelný systém
    - Cílem aplikace je poukázat na nereálné chování tepelného systému, kdy při linearizaci systému zanedbáváme i fyzikální vlastnosti kapalin. Též uživateli ukazuje vliv na chování systému při uvažování různé míry ztrát tepla v systému.

- **Úloha 4** – Linearizované elektrické systémy 1. a 2. řádu
    - Účelem aplikace je porovnat vliv výběru hodnot součástek na chování výstupu dvou podobných elektrických obvodů.

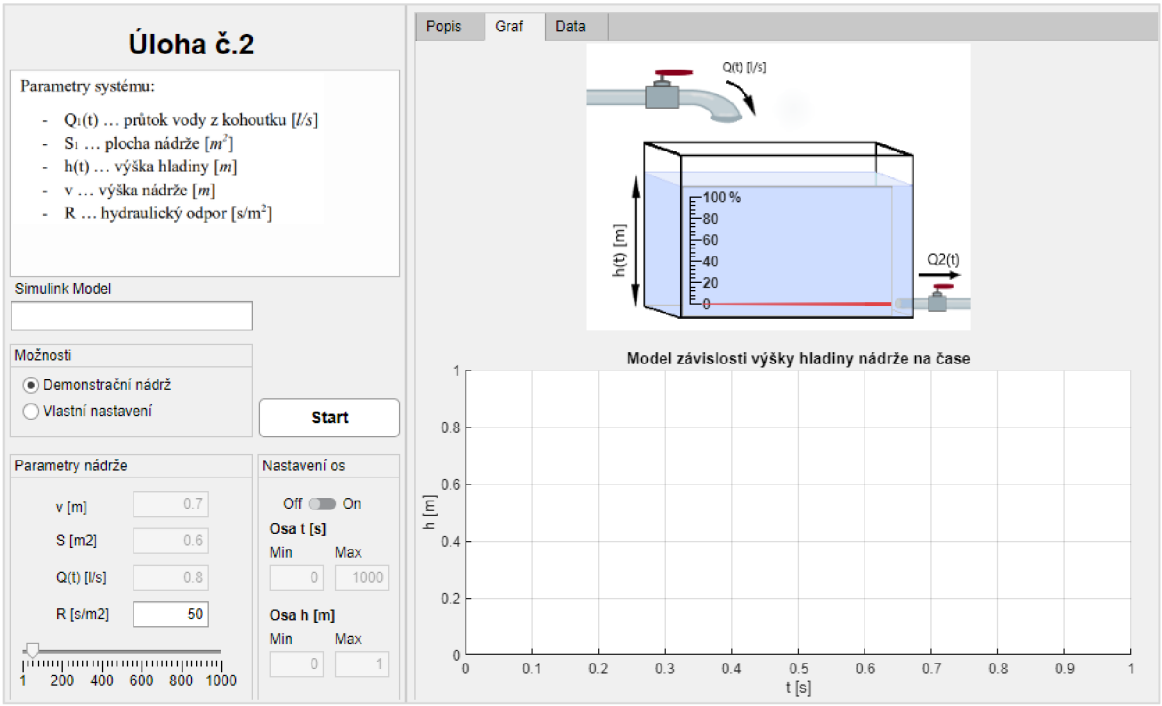
Porovnáním všech aplikací mezi sebou dokazujeme, že různé fyzikální systémy 1. řádu vykazují stejné charakteristiky, rozdílem jen zůstávají hodnoty os, které ovlivňují vstupní parametry systémů.

4.1 Návrh rozložení komponent v aplikacích

Při tvorbě nového GUI vybíráme ze tří typů rozvržení, ve kterém následně vytváříme celý vzhled aplikace viz kapitola 3.1. Pro naše aplikace použijeme rozvržení, které rozděluje aplikaci do dvou panelů (2-Panel App with Auto-Reflow), na pravý a levý panel.

V aplikaci využijeme levý panel pro ovládací prvky. Za ovládací prvky považujeme tlačítka, posuvníky, editovatelná pole textová nebo numerická, stavová tlačítka apod. Pravý panel naopak slouží pro výstup ze spuštěné simulace a pro popis příslušné úlohy.

Celé GUI je strukturováno tak, aby uživateli nabídlo co nejintuitivnější ovládaní bez dlouhého hledání příslušných parametrů, legend proměnných či spuštění simulace.



Obrázek 4.1.1: Návrh GUI aplikace – Úloha 2

Levý panel je rozložen do dvou pomyslných částí. V horní třetině je umístěn název aplikace s legendou použitých proměnných, aby byly pro uživatele stále viditelné.

Pod legendou se nachází textové pole, které vypíše název aktuálně spuštěného Simulink modelu. Uprostřed levého panelu je umístěno Start tlačítko, které spustí simulaci s defaultně nastavenými hodnotami. Vedle tlačítka Start je použit prvek Radio Button Group s názvem Možnosti. Tento prvek nabízí uživateli dvě možnosti, pracovat s předdefinovaným modelem nebo se přepnout na Vlastní nastavení a tím odemknout úpravu všech parametrů.

V dolní části levého panelu jsou umístěny dva bloky. První blok je pro zápis a předávání hodnot proměnných. K tomu využíváme prvek Edit Field. Každá proměnná systému je zastoupena jedním Edit Field. Druhý blok, s názvem Nastavení os, slouží pro úpravu os výstupního grafu a tedy usnadňuje uživateli práci s grafem.

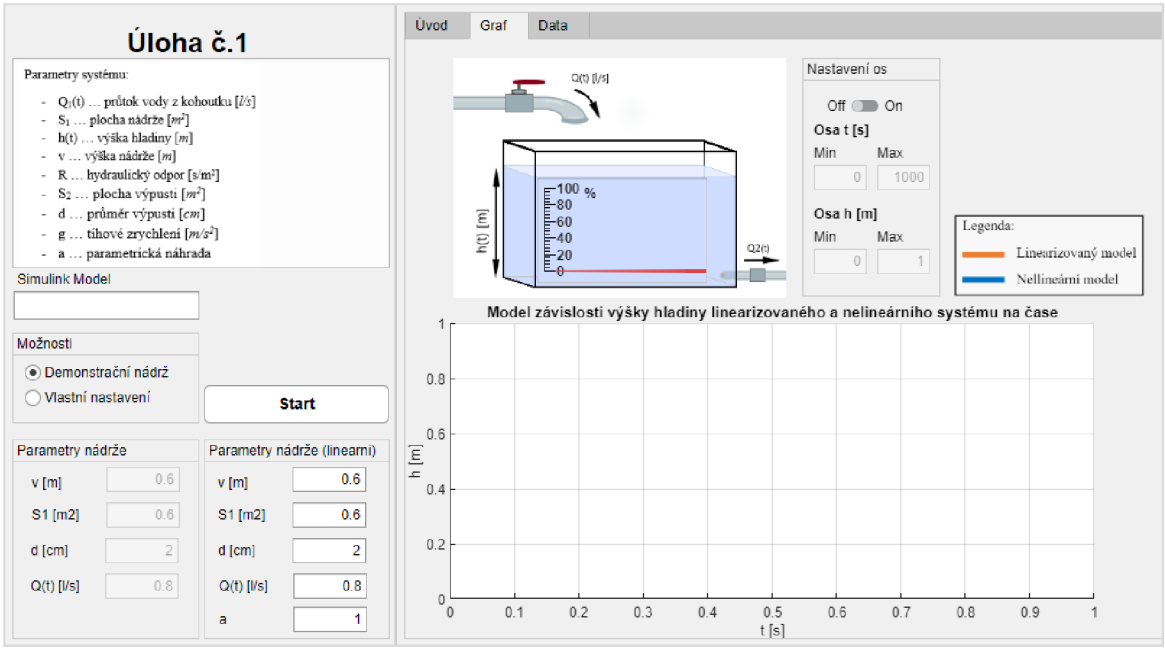
Celý pravý panel obsahuje prvek Tab Group. Tab Group na pravém panelu vytvoří záložky, tím v GUI vzniká více prostoru a aplikace zůstává přehledná pro uživatele.

Tab Group obsahuje tři záložky:

- **Úvod** – uživatel při otevření aplikace jako první vidí popis celé úlohy a stručné ovládání GUI
- **Graf** –záložka obsahuje graf se simulovaným výstup získaný ze Simulink modelu a obrázek systému
- **Data** – záložka obsahuje tabulku výstupních hodnot ze systému

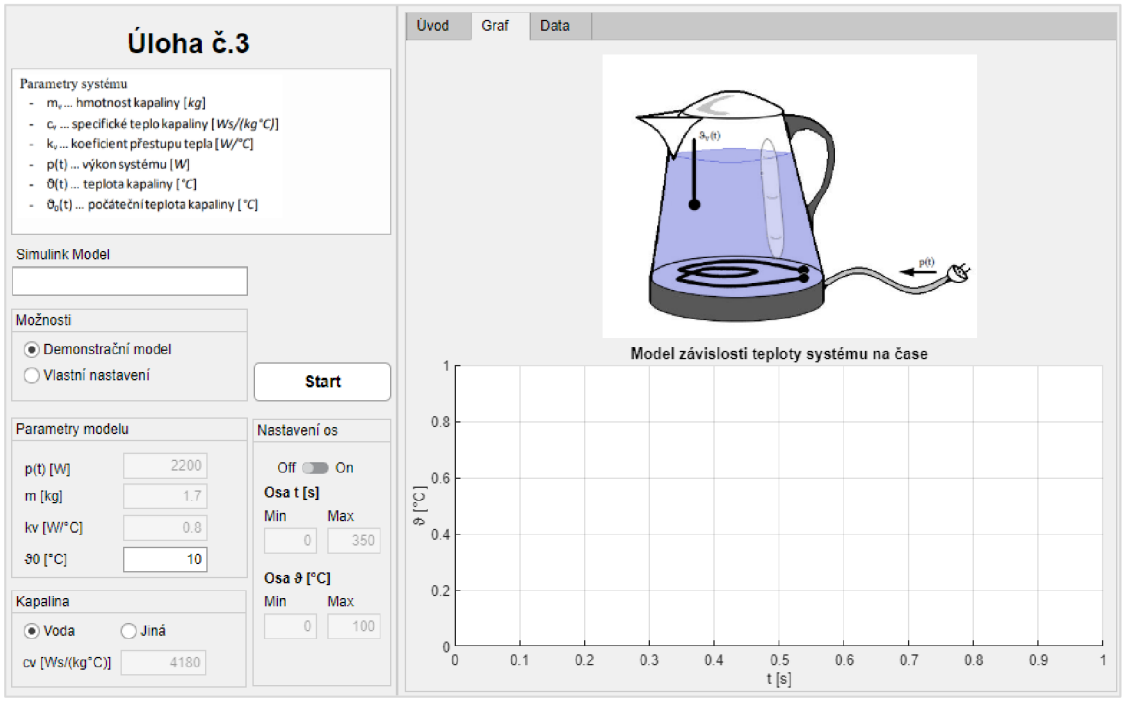
Návrh GUI úlohy 2 slouží jako šablona pro tvorbu ostatních návrhů aplikací. Rozdíly mezi jednotlivými GUI vznikají z důvodu potřeby rozšířit sadu ovládacích prvků individuálně pro jednotlivé aplikace. Jinak tento princip rozvržení GUI zůstalo zachováno.

Rozvržení GUI úlohy 1 jsme rozšířili o jednu sadu parametrů, protože v aplikaci porovnáváme dva na sobě nezávislé hydraulické systémy. Blok s druhou sadou parametrů umístíme na místo bloku Nastavení os, jak lze vidět na Obrázku 4.1.2. Jelikož na místě bloku Nastavení os je umístěn blok s parametry linearizovaného systému, přesuneme Nastavení os do pravého panelu nad graf vedle obrázku systému.



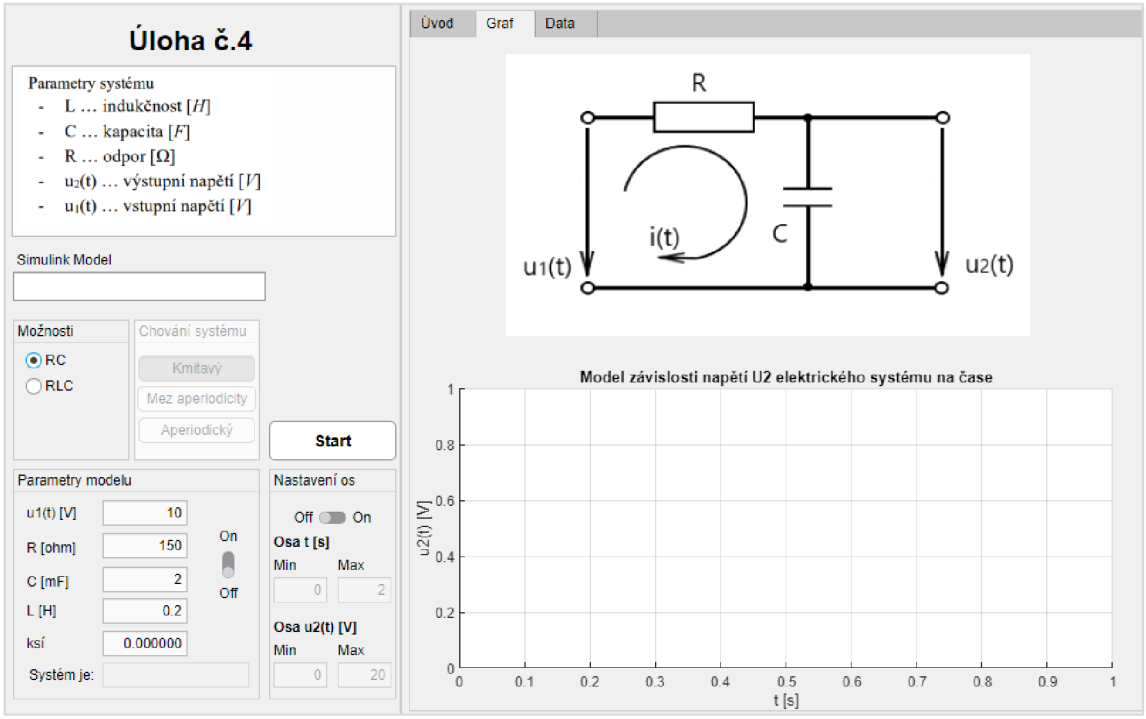
Obrázek 4.1.2: Návrh GUI aplikace – Úloha 1

GUI pro úlohu 3 jsme doplnili o jeden blok s názvem Kapalina, kde uživateli je nabídnuta možnost simulovat i jinou kapalinu než vodu. Blok je umístěn pod blokem s parametry viz Obrázek 4.1.4.



Obrázek 4.1.4: Návrh GUI aplikace – Úloha 3

Poslední aplikaci jsme rozšířili o blok s názvem Chování systému. Chování systému umožňuje uživateli vybrat chování systému 2. řádu, které se při spuštění simulace vykreslí. Blok Možnosti je modifikován tak, aby přepínal mezi systém RC a RLC. Prvek Vlastní nastavení jsme upravili na spínač On/Off, který je umístěn v bloku Parametry modelu viz Obrázek 4.1.3.



Obrázek 4.1.3: Návrh GUI aplikace – Úloha 4

5. IMPLEMENTACE APLIKACÍ V PROSTŘEDÍ MATLAB APP DESIGNER

V této kapitole v prostředí App Designer implementujeme vazby mezi komponenty v návrzích aplikací, které jsou popsány v předešlé kapitole. Též v kódu vytvoříme propojení mezi Simulink modely s příslušnou GUI aplikací. Propojení modelů Simulink a App Designeru je popsáno v podkapitole 3.2.

Programování v prostředí App Designer je založeno na principu objektového programování, kde v celém prostředí platí datová struktura app. Oproti starším verzím GUI se předpisy pro programování zjednodušili tím, že se upustilo od příkazů get a set a nyní se přiřazují hodnoty objektů napřímo [13].

V prostředí App Designer v oblasti Code view jsou uzamčeny části kódu, aby zamezili přepisu důležitých metod a funkcí, které by při špatné úpravě nebo smazání mohli znemožnit funkčnosti celého GUI. Tvůrce před začátkem programování musí vytvořit novou oblast, do které je povolen zápis.

Oblast s povolením zápisu lze vytvořit třemi způsoby, kdy si tvůrce na levé straně prostředí v sekci Code Browser vybere přidání Properties, Functions nebo vytvoří pro určitou komponentu Callbacks.

5.1 Implementace Úlohy 1

Návrh Úlohy 1 lze vidět v předešlé kapitole 4.1 na Obrázku 4.1.2. Návrh již obsahuje všechny potřebné komponenty k ovládání GUI, již jen na programujeme jejich chování, které vykonají při detekování akce v aplikaci.

Nejprve v sekci Desing View nastavíme výchozí hodnoty a rozsahy všech komponent typu Edit Field určené pro proměnné. Nastavení se provádí v Component Browser - > Inspector, též v této části lze nastavit jméno komponenty, velikost a styl písma, umístění prvku, interaktivita apod.

Po nastavení komponent přepneme do sekce s kódem a naprogramujeme propojení mezi Simulink modelem a GUI. V aplikaci používáme příkazy load_system a save_system. Syntaxe příkazů je následující:

```
%Callback spuštění aplikace
function startupFcn(app)
      load_system('Nadrz.slx');
end
% Callback zavření aplikace
function UIFigureCloseRequest(app)
      save_system('Nadrz.slx');
end
```

Příkaz load_system zapíšeme do callbacku spuštění celé aplikace, tedy model Simulink se načte současně s inicializací GUI. Příkaz save_system se provede v okamžiku, kdy se aplikace vypne, tedy při detekce callbacku zavření okna GUI. Název modelu, který je načítán do paměti aplikace, vložíme do jednoduchých uvozovek.

Pro vkládání hodnot z aplikace do modelu Simulink slouží příkaz set_param. Příkaz má mnoho možností zápisu [14]. V aplikaci používáme zápis, který vybere přímo blok v modelu a přepíše jeho hodnotu na žádanou GUI. Syntaxe příkazu je následující, za příkaz do jednoduchých uvozovek zapíšeme jméno modelu lomítko jméno bloku, následně se zapíše typ bloku, ve kterém se mění hodnota, nakonec vkládána hodnota v datovém typu string. Pro zápis hodnot do modelu jsme vytvořili vlastní funkci model(app).

```
function model(app)
      promenne();
      promenne_a();
      set_param ('Nadrz/Q1','Gain', num2str(app.q1));
end
```

V aplikaci používáme komponentu Axes, pro vykreslení přechodových charakteristik ze Simulink modelu, aby výčet dat fungovalo, musíme do modelu přidat blok Workspace, který ukládá výstupní data systému. Pro přístup používáme příkaz sim [15] a následně příslušná data vykreslujeme do komponenty Axes.

```
% Callback tlačítka Start
function StartButtonPushed(app, event)
      model();
      simul=sim('Nadrz.slx');
      plot(app.UIAxes,simul.out.Time,simul.out.Data,simul.out1.Time,…
          simul.out1.Data)
end
```

Když propojení mezi modelem Simulink a App Designerem funguje, začneme s implementací vlastnosti Vlastního nastavení modelu. Program přepneme do sekce Desing View. Zde všem komponentám, které uživatel aplikace muže měnit až po přepnutí na vlastní nastavení, vypneme funkci Editable, a tím nastavíme výchozí stav při zapnutí aplikace. V kódu následně zavedeme nové funkce promenne(app) a promenne_a(app), každá funkce je pro jednu sadu parametrů.

Ve funkci pro proměnné parametrizovaného systému jen přiřazujeme hodnoty z komponent příslušným globálním proměnný, které jsme definovali v Properties.

Funkce pro nelineární model je naprogramována tak, aby při přepnutí na Vlastní nastavení se povolilo editování komponent a hodnoty přepsané z Edit Fields se uložili do odpovídajících globálních proměnných v kódu. Při zpětném přepnutí na Demonstrační nádrž se naopak do Edit Fields přepíšou hodnoty definované pro demonstrační model a opět se tyto komponenty uzamknou.

```
function promenne(app)
   if app.Nadrz.Value                %Vybráno Demonstrační nádrž
      app.Qt.Editable='off';         %Komponenta není editovatelná
      app.q1=0.8;
      app.Qt.Value=app.q1;           %Do komponenty se nastaví 0.8

   elseif app.Free.Value             %Vybráno Vlastní nastavení
      app.Qt.Editable='on';          %Povolení editace komponenety
      app.q1=app.Qt.Value;

   end
end
```

Implementace Nastavení os v kódu definujeme funkcí osy(app). Pro úpravu rozsahů os ve vykresleném grafu musí se nejprve přepnout přepínač On/Off na On, tím se povolí editace polí pro zápis mezních hodnot. Volání funkce osy() je vloženo do čtyř callbacků komponent, ve kterých dochází ke změnám.

```
function osy(app)
   y=app.Hmin.Value;
   y2=app.Hmax.Value;
   x=app.Tmin.Value;
   x2=app.Tmax.Value;

   if x<x2 && y<y2                   %podmínka pro změnu hodnot na osách
      app.UIAxes.YLim=[y y2];
      app.UIAxes.XLim=[x x2];
   end
end
```

Celá aplikace pracuje následovně. Při prvotním spuštění se během inicializace aplikace načte Simulink model. Následně se čeká na akci uživatele. Spuštěním simulace se aktivuje callback tlačítka Start, který zavolá funkci model a nahraje aktuální proměnné do modelu. Následně se příkazem sim uloží data z modelu do pole simul a přejde se k vykreslení dat do grafu. Aplikace opět čeká na akci uživatele.

Pokud dojde k přepnutí Možností na Vlastní nastavení, callback bloku možností povolí editaci dosud zamčených parametrů. Při stisku tlačítka Start opět proběhne celý cyklus, který je popsán výše.

Jestliže přepínač v bloku Nastavení os nastavíme do pozice On, hodnoty na osách v komponentě Axes se nastaví na hodnotu zadanou. Změny hodnot os je vždy zapotřebí potvrdit stiskem klávesy Enter, aby se v programu vykonal callback příslušné mezní hodnoty.

Ukončením aplikace nastane uložení změn v modelu Simulink a smazání pomocné paměti dříve vyvolané příkazem load_system.

5.2 Implementace Úlohy 2

V kapitole 4.1 jsme navrhli rozložení všech komponent pro úlohu 2, viz Obrázek 4.1.1. GUI už jen naprogramujeme, a tím definujeme chování výsledné aplikace. V předešlé kapitole 5.1, je popsaná velká část programu této úlohy.

Oproti předešlé aplikaci toto GUI obsahuje pouze jednu sadu parametrů, proto ve funkci model(app) voláme jen jednu funkci na inicializaci proměnných. Sadu proměnných jsme doplnili o posuvník, který umožňuje rychleji měnit hodnotu parametru. Proměnnou R můžeme nastavovat pomocí dvou komponent, přičemž posuvník je omezen rozsahem od 1 do 1000. Rozsah je zvolen z důvodu čitelnosti posuvníku, v případě, že proměnná R zapsaná v Edit Field nabude vetší hodnoty než 1000, posuvník se ustálí na maximální hodnotě svého rozsahu.

```
% Callback posuvníku proměnné R
function SliderValueChange(app, event)
      app.R.Value=app.Slider.Value;
end

% Callback Edit Field proměnné R
function RValueChanger(app, event)
   if app.R.Value<1001
      app.Slider.Value=app.R.Value;
   else
      app.Slider.Value=1000;
   end
end
```

Implementovaná aplikace pracuje obdobně jako aplikace úlohy 1. Při spuštění aplikace příkaz load_system načte příslušný model Simulink. Po načtení, pokud uživatel nezmění hodnotu proměnné R a spustí simulaci, callback tlačítka Start zavolá funkci model(app) s parametry demonstrační nádrže s otevřeným výpustním ventilem, ty uloží a následně vykreslí graf závislosti do komponenty Axes.

Při změně parametru R, ať už komponentou Edit Field nebo posuvníkem, následným spuštěním aplikace se systém chová dle nastaveného parametru R (viz kapitola 2.1.1). Vždy po vykreslení grafu program čeká na další akci uživatele nebo na ukončení celé aplikace.

5.3 Implementace Úlohy 3

Úloha 3 vychází z návrhu GUI znázorněného na Obrázku 4.1.4 v kapitole 4.1. Implementovaný program aplikace vychází z programu úlohy 1, popsaném v podkapitole 5.1.

Program úlohy 1 upravíme tak, aby odpovídal aktuální aplikaci a rozšíříme o jednu funkci kapalina_nastav(app), pro nastavování koeficientu specifického tepla různých kapalin.

Funkci kapalina_nastav(app) voláme ve funkci model(app) a v callbacku bloku Kapalina, kdy při změně prvku z Vody na jinou kapalinu se zavolá funkce popsána níže.

```
%Callback bloku Kapalina
function KapalinaSelectionChanged(app, event)
   app.kapalina_nastav();
end
```

Při zavolání tohoto callbacku funkce povolí zápis do komponenty Edit Field proměnné c_v. V horní části levého panelu viditelnou legendu parametrů nahradí výpis vybraných kapalin s příslušným koeficientem přestupu tepla.

```
function kapalina_nastav(app)
   if app.Jina.Value
      app.Cv.Editable='on';
      app.c=app.Cv.Value;
      app.PoznamkaCv.Visible='on';  %výpis kapalin s koeficientem cv
      app.Legenda.Visible='off';    %schování legendy parametrů

   else
      app.c=4180;
      app.Cv.Value=app.c;
      app.Cv.Editable='off';
      app.PoznamkaCv.Visible='off'; % schování koeficientů cv
      app.Legenda.Visible='on';     % obnovení legendy parametrů
   end
end
```

Po spuštění aplikace se program vykonává následovně, proběhne inicializace GUI, načtení modelu do paměti a program vyčkává na akci uživatele. Při změně v bloku Kapalina se uživateli odemkne přístup k zápisu hodnoty proměnné c_v. Po spuštění programu tlačítkem Start simulace vykreslí graf závislosti pro zadanou kapalinu.

Stejně jako u předešlých aplikací po vykreslení grafu program čeká na další akci uživatele nebo na ukončení celé aplikace.

5.4 Implementace Úlohy 4

Návrh úlohy 4 je popsán v podkapitole 4.1 a obsahuje všechny potřebné komponenty viz Obrázek 4.1.3. Program této úlohy z většiny vychází z programu úlohy 1, kde probíhá propojení jen s jedním modelem Simulink, kdežto úlohu 4 je zapotřebí propojit se dvěma modely.

První model obsahuje elektrický systém 1. řádu a druhý systém 2. řádu. Program při inicializaci aplikace načte dva na sobě nezávislé modely Simulink. Celý program musí být doplněn o podmínky rozhodující, který model je zapotřebí simulovat.

V program jsme doplnili o dvě nové funkce, které využíváme při simulaci systému 2. řádu. První funkce slouží k výběru chování demonstračního modelu, kdy

uživatel si vybere z možností– Kmitavý, Mez aperiodicity nebo Aperiodický systém. Funkci voláme v callbacku objektu Chování systému.

```
function chovani(app)
      if app.Kmitavy.Value
            app.r=5;
            app.R.Value=app.r;
      elseif app.Mezaperiodicity.Value
            app.r=20;
            app.R.Value=app.r;
      elseif app.Aperiodicky.Value
            app.r=50;
            app.R.Value=app.r;
      end
end
```

Druhá funkce vypocetsystemu(app), vykonává výpočet hodnoty poměrného tlumení ξ, aktuálně simulovaného modelu a na základě této hodnoty určuje chování systému. Funkce je především určena pro možnost vlastního nastavení, kdy uživatel změní hodnoty proměnných, a tím i chování systému. Po načtení nových hodnot proměnných a výpočtu poměrného tlumení vypíše do komponenty Edit Field, jak chování systému, tak i vypočtenou hodnotu ξ.

```
function vypocetsystemu(app)
      odlc=sqrt(app.lc);            %výpočet poměrného tlumení
      app.ksi=app.rc/(2*odlc);
      app.KSI.Value=app.ksi;

%Rozhodovací logika chování systému
      if 0 <app.ksi && app.ksi<0.9999999
            app.SystemTyp.Value='Kmitavý';
      elseif 0.9999999<app.ksi && app.ksi<1.0000001
            app.SystemTyp.Value='Mezní aperiodický';
      else
            app.SystemTyp.Value='Aperiodický';
      end
end
```

Aplikace při spuštění načítá do pamětí oba modely Simulink, defaultně má GUI nastavený RC model. Pokud uživatel chce simulovat RLC, musí přepnou v bloku Možnosti na model RLC. Při přepínání mezi jednotlivými modely se v záložce Graf mění schéma modelu.

Je-li nastavený model RLC, odemkne se nastavení Chování systému, jestliže se současně povolí vlastní nastavení proměnných, Chování systému se opět uzamkne.

Ukončením aplikace nastane uložení změn v modelech Simulink a smazání pomocných pamětí dříve vyvolaných příkazem load_system.

6. DEMONSTRACE FUNKČNOSTI APLIKACÍ

V této kapitole otestujeme funkčnost všech čtyř aplikací, které jsou implementovány v kapitole 5. Pro spuštění aplikací využijeme hlavní okno prostředí MATLAB, které se otevře vždy při spuštění prostředí.

K jednotlivým aplikacím přistupujeme z příslušné složky v MATLABu. Ve složce klikneme pravým tlačítkem myši na soubor s příponou .mlapp a klikneme v na Run v menu. Aplikace se spouští přímo z prostředí, aby došlo ke změně složky s výchozími daty.

6.1 Test aplikace Úlohy 1

V prostředí MATLAB spustíme aplikaci Uloha1.mlapp. První vidíme na pravé straně stručný popis úlohy, ovládání GUI a modely Simulink obou hydraulických systému. V levé části lze vidět soupis parametrů a pod tím ovládací prvky aplikace.

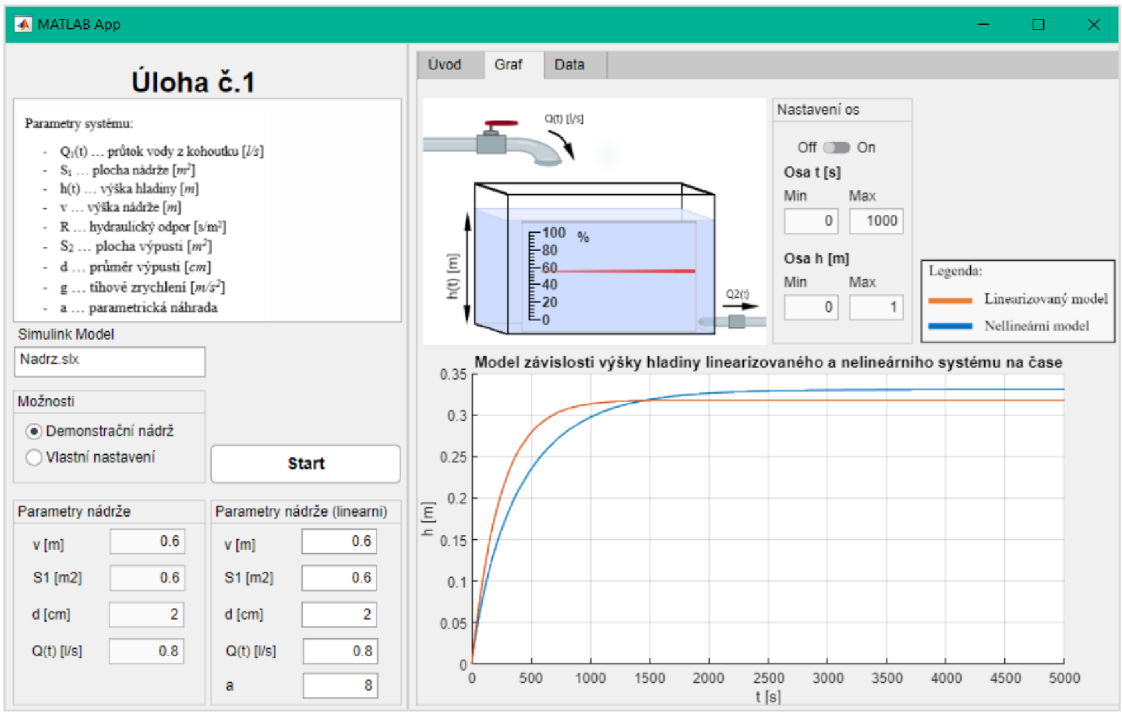
Zapneme simulaci tlačítkem Start bez změny nastavení a přepneme záložku Úvod na záložku Graf, zde vidíme vykreslené dvě přechodové charakteristiky. Výstup nelineárního systému přesáhl 50 % plnosti nádrže. Procentní vyjádření plnosti nádrže lze též vidět na obrázku systému, kde ustálený stav hladiny znázorňuje červený ukazatel. Výstup parametrizovaného systému dosáhl maxima.

V druhé simulaci zkusíme přechodovou charakteristiku linearizovaného systém přiblížit k nelineární, změnou hodnoty parametru a z jedné na osm. Linearizovaným systémem jsme se dostali pod charakteristiku nelineárního systému viz Obrázek 6.1.1.

Následně otestujeme, zda je správně implementována možnost nastavení os. Přepínač přepneme do pozice On a vzápětí vidíme, jak se osy grafu upravili podle předdefinovaných mezí. Nastavení os je opatřeno proti zadání vyšší hodnoty na dolní hranici. Při pokusu o nastavení časové osy vyšší nebo stejnou hodnotou jako na hranici horní, graf zůstal beze změny. Při opětovném přepnutí přepínače do pozice Off se graf vrátí do původního nastavení.

V možnostech přepneme z Demonstrační nádrže na možnost Vlastního nastavení modelu, tím se nám povolil zápis do proměnných nelineární nádrže. Změníme hodnoty v proměnných, všechny hodnoty se přepsaly. Při zadání hodnoty mimo povolený rozsah proměnné, aplikace nedovolila zápis a ukázala, v jakém rozsahu lze zadat danou proměnou. Tuto schopnost obsahují i ostatní aplikace.

Nakonec porovnáme výsledky dosažené v aplikaci s výsledky v kapitole 2.1.1. Výsledky získané ze simulace v prostředí Simulink zcela odpovídají výsledným závislostem vykreslených v GUI, při zadáví stejných parametrů.



Obrázek 6.1.1: Test aplikace Úlohy 1

6.2 Test aplikace Úlohy 2

Spustíme aplikaci Uloha2.mlapp. Obdobně jako u Úlohy 1 první vidíme popis dané úlohy a model hydraulického systému, který je implementován v prostředí Simulink. Na levé straně je umístěn soupis parametrů a ovládací prvky aplikace.

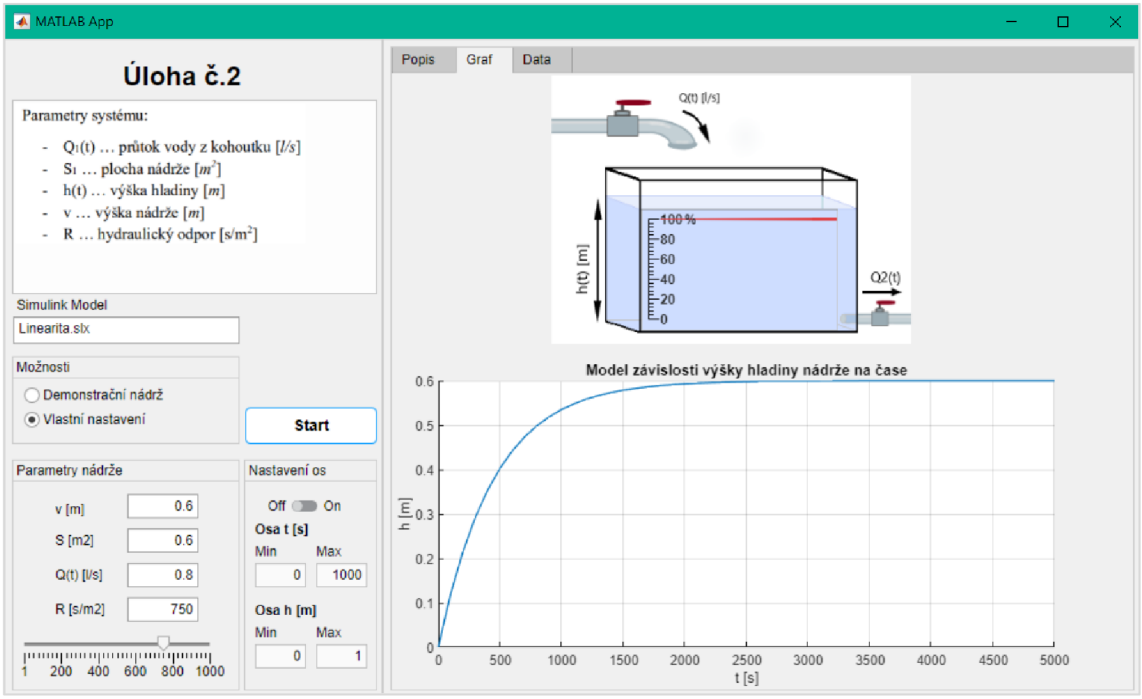
Přepneme se ze záložky Popis na Graf a zapneme simulaci s defaultním nastavením proměnných. Vykreslená přechodová charakteristika hydraulického systému dosahuje přibližně 7 % plnosti nádrže z důvodu nastavení hydraulického odporu na R=50 s/m^2. Procentní vyjádření výšky hladiny v nádrži lze opět vidět nad grafem v obrázku systému.

Změníme-li hodnotu proměnné R na posuvníku, vidíme následné propsání na příslušné komponentě a naopak. Hydraulický odpor nastavíme na R=650 s/m^2 a spustíme druhou simulaci a tím otestujeme, zda aplikace reaguje na změny. Výsledná výška hladiny vystoupala téměř do 80 %.

Nyní přepneme na možnost vlastního nastavení a upravíme hodnoty všech proměnných. Jelikož se nám povedlo provést zápis do všech proměnných, implementace Vlastního nastavení je správná.

Otestování implementace nastavení os provedeme přepnutím přepínače z pozice Off na On. Reakci vidíme vzápětí, když se graf přizpůsobil přednastaveným hodnotám. Upravíme-li jakoukoliv mez, není-li dolní mez větší než horní, graf upraví své rozsahy. Vypnutím této funkce přepínačem se graf vrátí do výchozího nastavení.

Porovnáme-li výsledek získaný v aplikaci s výsledkem dosaženým při implementaci systému v prostředí Simulink v kapitole 2.1.1, vidíme, že aplikace GUI zcela odpovídá.



Obrázek 6.2.1: Test aplikace Úlohy 2

6.3 Test aplikace Úlohy 3

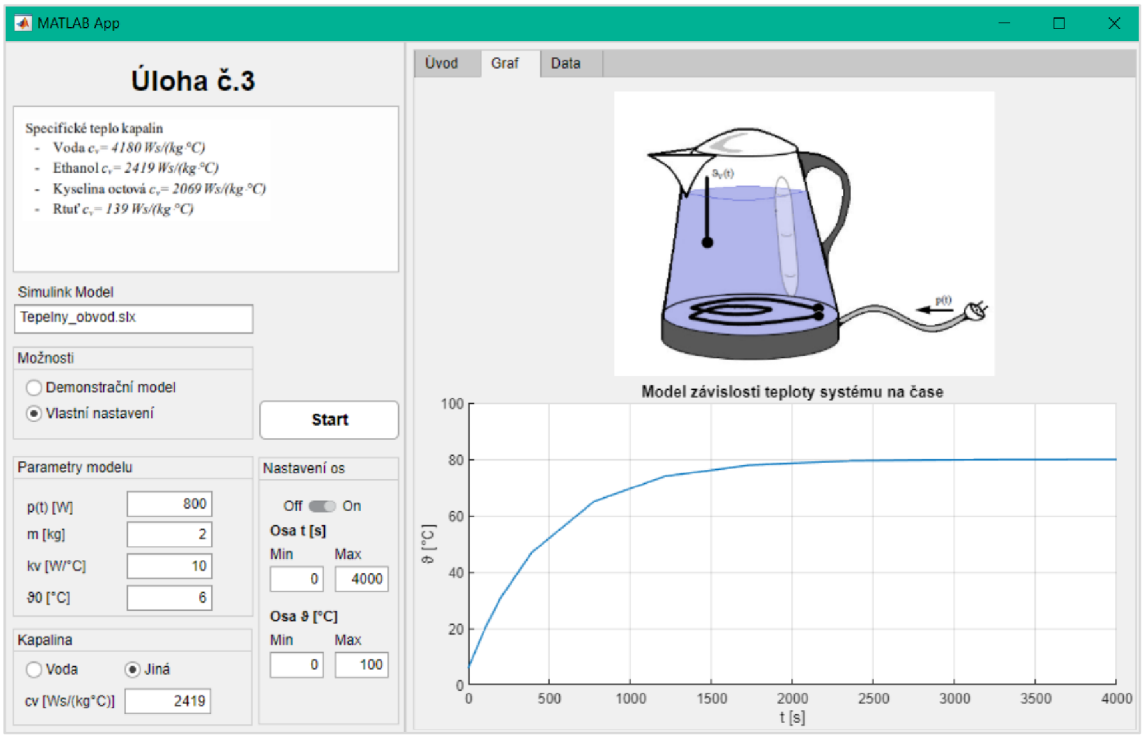
Po zapnutí aplikace Uloha3.mlapp v prostředí MATLAB opět vidíme v pravé části popis aplikace se stručným ovládáním popsaném v bodech a model tepleného systému. Na levé straně legendu parametrů využívaných v aplikaci a ovládací prvky.

Spustíme simulaci s předdefinovanými parametry a přepneme se do záložky Graf, kde vidíme přechodovou charakteristiku tepelného systému. Maximální teplota systému dosahuje hodnoty ϑ= 2758.7 °C, odůvodnění tohoto jevu je popsáno v kapitole 2.1.2.

V aplikaci není přímo vytvořená komponenta pro zadání bodu varu kapaliny. Funkce, která by detekovala bod varu v aplikaci, zastupuje Nastavení os. Po zapnutí bloku Nastavení os, se graf defaultně nastaví na bod varu vody a úměrně k tomu zkrátí časovou osu.

Následně otestujeme implementaci možnosti změny kapaliny, kde v bloku Kapalina přepneme z volby Voda na Jiná. Po akci se v levé horní části změnila legenda parametrů na výpis některých kapalin s příslušnými koeficienty specifického tepla kapalin a odemkla komponentu pro zápis těchto hodnot. Pokud opět přepneme na možnost Voda v komponentě specifické teplo, nastaví se hodnota c_v =4180 Ws/(kg°C), tedy na hodnotu odpovídající vodě a legenda parametrů se vrátí zpět.

Otestování funkce vlastního nastavení provedeme obdobně jako u předchozích aplikací. Přepneme z Demonstračního modelu na možnost Vlastního nastavení a vyzkoušíme změnit všechny parametry v bloku Parametry modelu. Změna se povedla u všech proměnných. Při zpětném přepnutí na Demonstrační model se do prvcích opět zapíší výchozí hodnoty. Všechny implementované funkce reagují podle očekávání. Můžeme tedy říct, že aplikace pracuje bezchybně.



Obrázek 6.3.1: Test aplikace Úlohy 3

6.4 Test aplikace Úlohy 4

Aplikaci Uloha4.mlapp spustíme v prostředí MATLAB. Po načtení celého GUI vidíme na levé straně legendu parametrů systému a ovládací prvky pro oba modely Simulink. Na pravé straně lze vidět stručný úvod do problematiky aplikace a Simulink modely.

Spuštěním simulace a přepnutím do záložky Graf otestujeme, zda aplikace je propojena se Simulink model RC systému. Přechodová charakteristika se vykreslila, přejdeme na test druhého elektrického systému, kterým je RLC systém. Přepnutím v možnostech na RLC obvod se povolil panel s názvem Chování systému, který je nastaven na kmitavý systém. Spustíme simulaci. Přechodová charakteristika vykazuje chování kmitavého systému. Následně zkoušíme i zbývající dvě přechodové charakteristiky, rovněž úspěšně.

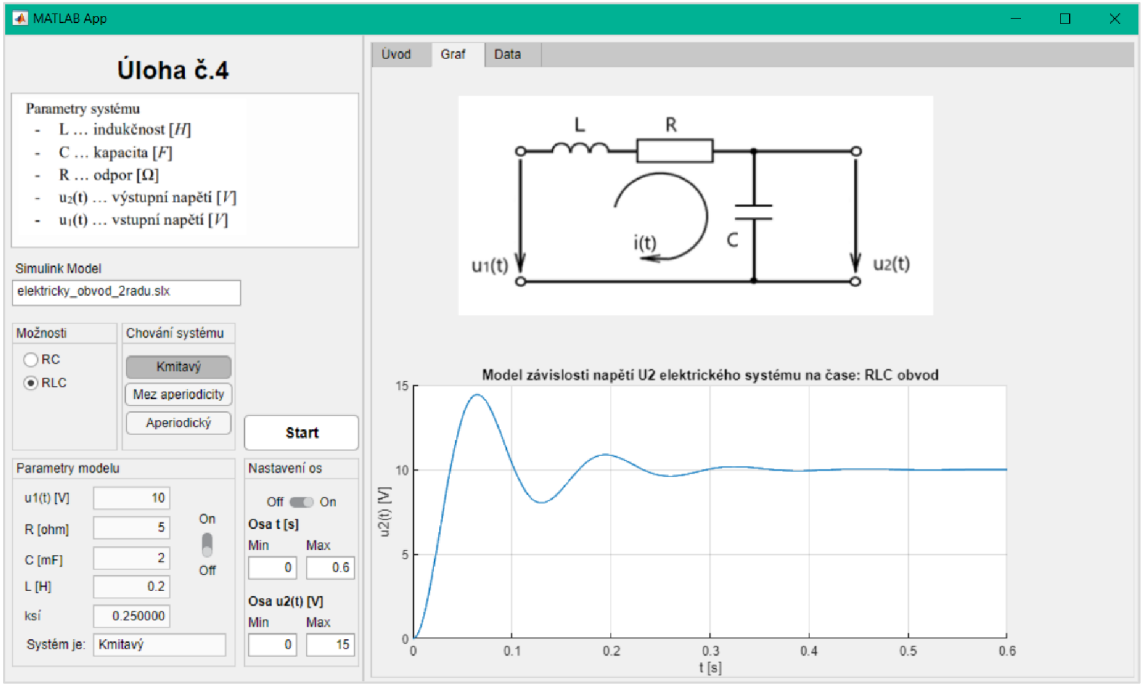
V této aplikaci je možnost vlastního nastavení přesunuta do bloku parametrů modelu formou přepínače. Přepnutím do pozice On s ponecháním RLC systému se opět zamkne panel Chování systému a všechny komponenty proměnných se naopak odemknou.

Změníme hodnoty proměnných a zapneme simulaci. Po stisku tlačítka Start se upravila hodnota proměnné ξ dle výpočtu programu a chování systému se vypsalo do komponenty Systém je:.

Přepneme-li během vlastního nastavení na systém RC uzamkneme tím proměnou L, která není využívána v RC obvodu. Ostatní proměnné zůstávají editovatelné do vypnutí vlastního nastavení.

Nastavení os v aplikaci pracuje na stejném principu jako v předchozích aplikacích. Otestování funkce je následující, přepneme přepínač do polohy On, odemkneme meze os a současně se graf přizpůsobí předdefinovanému nastavení. Po přepnutí zpět se graf vrátí do výchozího nastavení.

Nakonec porovnáme výsledky dosažené v aplikaci s výsledky modelů Simulink v podkapitole 2.1.3. a kapitole 2.2. Získané přechodové charakteristiky z GUI zcela odpovídají výsledným závislostem simulovaných v prostředí Simulink.



Obrázek 6.4.1: Test aplikace Úlohy 4

ZÁVĚR

Výsledkem bakalářské práce jsou čtyři funkční aplikace, které simulují chování vybraných reálných systémů na základě přechodových charakteristik a jejich dynamických vlastností. Reálné systémy zastupuje hydraulický (nádrž s výpustí), tepelný (rychlovarná konvice) a elektrický systém 1. a 2. řádu (RC a RLC obvod).

V kapitole Výběr reálných systémů teoreticky popisujeme jednotlivé systémy ohledně jejich základních dynamických vlastností a matematickým vyjádřením diferenciálních rovnicích, které využíváme pro implementaci do Simulink modelů v navazující kapitole. U systému prvního řádu odvozujeme z operátorového přenosu systému *F(p)* příslušné časové konstanty.

Po implementaci do prostředí MATLAB Simulink všech systémů prvního řádu ověřili, zda teoretické hodnoty časových konstant odpovídají hodnotám, které odečítáme ze simulovaných přechodových charakteristik. Teoretická hodnota časové konstanty hydraulického systému je rovna 450 vteřin a odečtená hodnota $T_{h(t)} = 450.2$ s. Vypočtená časová konstanta tepelného systému se rovna 8882.5 vteřin, zatím co odečtená hodnota z výstupního grafu odpovídá $T_{\vartheta(t)} = 8882.86$ s. Poslední systém prvního řádu, který ověříme je elektrický RC systém s hodnotou časové konstanty 0.3 vteřiny a naměřenou $T_{u2(t)} = 0.2983$ s. Všechny časové konstanty jsou s odchylkou, kdy nejmenší odchylka nastává u RC obvodu a je rovna 0.0017s, jsme tedy v řádech milisekund. Rozdíly mezi časovými konstantami mohly být zapříčiněny nepřesnou volbou polohy přímky pro nalezení časové konstanty, v práci násobíme ustálený stav výstupního systému 63 %. Přesné vyjádření této hodnoty je 63.212 %.

V kapitole tři se seznamujeme s vývojovým prostředím App Designer, který slouží pro tvorbu jednoduchých GUI aplikací přímo v prostředí MATLAB a s jednou z možností, jak propojit prostředí Simulink s App Designer i jaké příkazy se k tomu využívají.

V následující kapitole utváříme využití jednotlivých aplikací, jejich vizuální stánku, ať jednotlivé komponenty jsou umístěny systematicky a přehledně pro uživatele. To vedlo k rozložení GUI do dvou panelů, kde levý panel slouží pro ovládací prvky a výpis proměnných, které jsou uživateli po celou dobu na očích, dále pravého panelu pro popis a ovládání úlohy, též pro vykreslování výstupních přechodových charakteristik.

Implementace a testování aplikace probíhala současně pro správné naprogramování chování aplikace. Průběžně při psaní kódu jsme spouštěli aplikaci, abychom otestovali, zda aktuálně implementovaná funkce reaguje dle očekávaní, nebo je zapotřebí udělat změnu v programu. Výsledkem jsou čtyři plně funkční aplikace.

LITERATURA

[1] JIRGL, Miroslav. *Signály a systémy*. Brno, 2019. Prezentace. VUT v Brně.

[2] JURA, Pavel. *Signály a systémy-část 2.* Brno, 2017. Skriptum. VUT v Brně.

[3] *Teorie řízení* [online]. Kutná Hora: VOŠ a SPŠ, 2016 [cit. 2021-01-02]. Dostupné z: <http://www.edumat.cz/texty/teorie_rizeni.pdf>

[4] KOTLÍK, Bohumír. *Matematické, fyzikální a chemické tabulky pro SŠ a nižší ročníky víceletých gymnázií*. 2. vyd. Praha: Fragment, 2011. ISBN 978-802-5312-278.

[5] *Aproximace přenosových funkcí promocí přechodové charakteristiky* [online]. České Budějovice: Jihočeská univerzita, 2019 [cit. 2022-04-20]. Dostupné z: <http://home.pf.jcu.cz/~kyklop/SERYM/automatizace/jer/Kap09/Kap_09.htm>

[6] *Aproximace proporcionální soustavy se setrvačností 1. řádu* [online]. Ostrava: VŠB, 2006 [cit. 2022-04-20]. Dostupné z: <http://books.fs.vsb.cz/Identifikace/str/metody.htm#ma1>

[7] *MATLAB GUI: Create apps with graphical user interfaces in MATLAB* [online]. USA: MathWorks, 2018 [cit. 2020-12-10]. Dostupné z: <https://www.mathworks.com/discovery/matlab-gui.html?s_tid=srchtitle>

[8] *Simulink — Functions* [online]. USA: MathWorks, 2018 [cit. 2022-04-29]. Dostupné z: <https://www.mathworks.com/help/simulink/referencelist.html?type=function&s_tid=CRUX_topnav>

[9] *Load_system* [online]. USA: MathWorks, 2018 [cit. 2022-04-29]. Dostupné z: <https://www.mathworks.com/help/simulink/slref/load_system.html>

[10] *Open_system* [online]. USA: MathWorks, 2018 [cit. 2022-04-29]. Dostupné z: <https://www.mathworks.com/help/simulink/slref/open_system.html>

[11] *Save_system* [online]. USA: MathWorks, 2018 [cit. 2022-04-29]. Dostupné z: <https://www.mathworks.com/help/simulink/slref/save_system.html>

[12] *Close_system* [online]. USA: MathWorks, 2018 [cit. 2022-04-29]. Dostupné z: <https://www.mathworks.com/help/simulink/slref/close_system.html>

[13] *30. Matlab – GUI 1. začíname* [online]. Bratislava: Posterus, 2011 [cit. 2021-12-14]. Dostupné z:<http://www.posterus.sk/?p=10077&fbclid=IwAR2M-A6pOT33nQEZBtLA81lcDdWbm4kZB1zi-BdnBQgglAQ-Ya3PW5BzLt0>

[14] *Set_param* [online]. USA: MathWorks, 2018 [cit. 2022-04-29]. Dostupné z: <https://www.mathworks.com/help/simulink/slref/set_param.html>

[15] *Sim* [online]. USA: MathWorks, 2018 [cit. 2022-04-29]. Dostupné z: <https://www.mathworks.com/help/simulink/slref/sim.html>

SEZNAM PŘÍLOH

PŘÍLOHA A – Zdrojový soubor aplikace Úloha 1- je uložen na přiloženém flash disku
PŘÍLOHA B – Zdrojový soubor aplikace Úloha 2- je uložen na přiloženém flash disku
PŘÍLOHA C – Zdrojový soubor aplikace Úloha 3- je uložen na přiloženém flash disku
PŘÍLOHA D – Zdrojový soubor aplikace Úloha 4- je uložen na přiloženém flash disku