UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2013 Lukáš Vladyka

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Simulátor hydraulické soustavy

Lukáš Vladyka

Bakalářská práce

2013

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Lukáš Vladyka**
Osobní číslo:	**I10482**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Řízení procesů**
Název tématu:	**Simulátor hydraulické soustavy**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování:

Anotace:
Student v programovacím jazyku JAVA vytvoří model složitější hydraulické soustavy (např. série nádrží se spojenými dny), tak aby tento model byl použitelný jako simulovaná laboratorní úloha.š
Pracovní postup:
Úvodem bude třeba na základě matematicko-fyzikální analýzy sestavit bilanční rovnice popisující systém. Dalším krokem bude sestavení modelu ve vývojovém prostředí Matlab. Získaný model bude sloužit nejprve ke stanovení "rozumných" hodnot některých parametrů (rozměry nádrží, rozměry výtokových otvorů, ...) a posléze k verifikaci konečných výsledků. Stěžejní částí práce pak bude vytvoření tohoto modelu v jazyku JAVA tak, aby simuloval chování hydraulické soustavy v reálném čase. Vstupy a parametry modelu bude možno zadávat přes GUI nebo ze souboru, sledované stavové veličiny a výstupy bude možno zobrazovat číselně, graficky či ukládat do souboru. Chování modelu bude porovnáno s modelem dříve vytvořeným v Matlabu. Student odevzdá textovou část práce zpracovanou podle příslušných doporučení a norem (zejména ČSN ISO 690) a médium s oběma vytvořenými modely. Textová část práce bude obsahovat rešerši věnovanou modelování technologických procesů, odvození matematicko-fyzikálního modelu, popis tvorby modelů v Matlabu a v Javě a uživatelskou příručku.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

DUŠEK, F.; HONC, D. Matlab a Simulink - úvod do používání. Pardubice : Univerzita Pardubice, 2005. 172 s. ISBN 80-7194-475-0.
BALÁTĚ, J. Automatické řízení. Praha : BEN, 2004. 663 s. ISBN 80-7300-148-9.

Vedoucí bakalářské práce: **Ing. Petr Doležel, Ph.D.**
Katedra řízení procesů

Datum zadání bakalářské práce: **6. listopadu 2012**
Termín odevzdání bakalářské práce: **10. května 2013**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Daniel Honc, Ph.D.
vedoucí katedry

V Pardubicích dne 29. března 2013

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 14. 8. 2013

Lukáš Vladyka

ANOTACE

Práce se zaměřuje na sestavení hydraulické soustavy a bilančních rovnic popisující soustavu. Za pomoci simulace vhodným softwarem bylo zjištěno nejlepší nastavení hodnot pro ustálení soustavy, které byly použity pro další práci. Hlavní náplní práce bylo použití vhodné numerické metody k řešení bilančních rovnic, následná simulace v GUI JAVA a na závěr porovnání a vysvětlení odchylek grafických výstupů obou simulací.

KLÍČOVÁ SLOVA

Bilanční rovnice, hydraulická soustava, simulace, numerická matematika

TITLE

Simulator of Hydraulic System

ANNOTATION

The work focuses on assembly hydraulic system and balance equations describing the system. With the help of an appropriate simulation software was found the best setting for stable system that was used for further work. The main contents of the work was use a suitable numerical methods for solving balance equations, the subsequent simulation in JAVA GUI and at the conclusion comparison and explanation of deviations graphical output both simulations.

KEYWORDS

Balance equation, hydraulic system, simulation, numerical Mathematics

Obsah

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK A ZNAČEK

GUI	Graphical User Interface
SIMULINK	Simulation and Model-Based Design
MATLAB	MATrix LABoratory
SQL	Structured Query Language
AWT	Abstract Windowing Toolkit
JFC	Java Foundation Classes
SWT	Standard Widget Toolkit
IDE	Integrated Development Environment
JDK	Java Development Kit
API	Application Programming Interface

ÚVOD

Úkolem této bakalářské práce je navržení a simulace hydraulické soustavy. Navržení hydraulické soustavy záleží na vlastním uvážení, tj. kolik nádrží se použije a jak budou umístěny. Pro řešení bakalářské práce jsou vybrány tři nádrže pro napouštění vodou. Umístění je zvoleno tak, že dvě nádrže jsou umístěny a propojeny vedle sebe a třetí nádrž se nachází pod jednu ze dvou nádrží. Přitom napouštění je zvoleno tak, že přítok je zaveden pouze do jedné nádrže, která má označení hodnotou jedna.

Pro další postupy je hydraulická soustava na základě matematicko-fyzikální analýzy popsána bilančními rovnicemi. Podle známých vzorců a rovnic jsou sestaveny diferenciální rovnice. Ty se následně používají pro další účely. V tomto případě slouží pro sestavení rovnic simulačního softwaru nazvaný SIMULINK, který je nadstavbou vývojového prostředí MATLAB. V uvedeném softwaru je podle rovnice sestaveno schéma a následně provedena simulace, kde výstupem je graf s vyobrazením závislosti hladiny nádrží na čase. Pomocí pokusů a omylů simulací se určí hodnoty a rozměry napouštění, průřezu výpustí a průměrů nádrží.

Získané hodnoty, popsané v předešlém bloku, jsou dále použity pro simulaci v jazyce JAVA pomocí rozhraní GUI. V tomto programovacím jazyce je vytvořen panel s možností nastavení hodnot nádrží, které byly zjištěny v předešlé simulaci. Přitom je nutné vyřešit nedostatek jazyku JAVA a to, že neumí pracovat s diferenciálními rovnicemi. Pro tento účel se používá metoda Runge-Kutta, která je jednou z nejpřesnějších numerických metod pro řešení diferenciálních rovnic popisujících stávající hydraulickou soustavu. Tato metoda se aplikuje právě do jazyku JAVA a následně pomocí zjištěných hodnot je soustava znovu nasimulována.

Na závěr a podstatě řešení bakalářské práce je posouzení výsledných grafů nebo hodnot simulací ze softwaru SIMULINK a JAVA, kde se porovnává odchylka simulací a vysvětlení proč tomu tak je.

Do přílohy je vložena část kódu, která slouží jako ukázka práce s numerickou metodou řešící diferenciální rovnice hydraulické soustavy.

1 Teoretická část

Úvodem bakalářské práce je nutné přiblížit a vysvětlit uživateli určité teoretické předpoklady a zákony. Určitým teoretickým předpokladem jsou diferenciální rovnice popisující hydraulickou soustavu, se kterými se pracuje v celé bakalářské práci a jedná se o jednu z nejdůležitějších částí práce. Pro zpracování a ověření správnosti diferenciálních rovnic soustavy se dá využít různých nástrojů. Konkrétně je tu využit Simulink, který je součástí výpočetního programu MATLAB. Práce je však zaměřená i na simulaci v jazyku JAVA. Pro implementaci do jazyku JAVA se musí diferenciální rovnice upravit vhodnou numerickou metodou. Za tímto účelem se používá metoda Runge-Kutta. Dále je důležité si popsat určité zákonitosti, které ovlivňují děj a průběh celé práce. Teoretické předpoklady a zákony jsou dále rozvedeny níže do samostatných bloků.

1.1 MATLAB

Název MATLAB vznikl z anglického matrix laboratory. MATLAB byl napsán, aby poskytoval jednoduchý přístup k matematickým knihovnám vyvinutý v projektech LINPACK a EISPACK. Byl původně určen pro operační systémy UNIX a tato okolnost dodnes projevuje ve velmi jednoduchém základním komunikačním rozhraní – příkazové řádce [3].

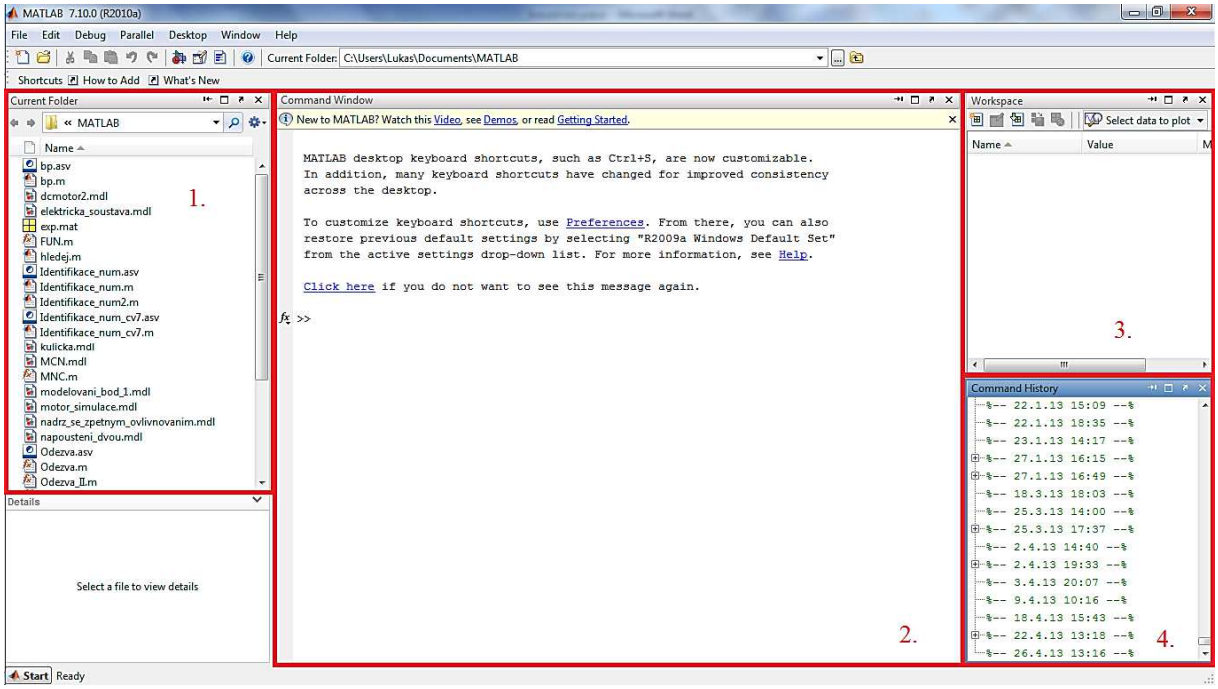
MATLAB je vysoce výkonný jazyk pro technické výpočty. Integruje výpočty, vizualizaci a programování do jednoduše použitelného prostředí, kde problémy i řešení jsou vyjádřeny v přirozeném tvaru. Jde o interaktivní systém, jehož základním datovým typem je dvourozměrné pole, bez nutnosti deklarovat rozměry. Tato vlastnost spolu s množstvím zabudovaných funkcí umožňuje relativně snadné řešení mnoha technických problémů, speciálně takových, které vedou na vektorovou či maticovou formulaci, v mnohem kratším čase než řešení v klasických jazycích jako je C. Typické oblasti použití jsou [3]:

- Inženýrské výpočty
- Vývoj algoritmů
- Modelování, simulace a vývoj prototypů
- Analýza dat a jejich vizualizace
- Inženýrská grafika
- Vývoj aplikací včetně tvorby grafického uživatelského rozhraní

V univerzitním prostředí jde o standardní nástroj využívaný ve výuce matematiky a inženýrských oborů. V průmyslu je využíván jako vysoce efektivní nástroj pro výzkum, vývoj

i analýzu dat. Ve srovnání s jinými obdobnými produkty je MATLAB blíže programovacímu jazyku. Nemá tolik předpřipravených komplexních matematických funkcí jako např. Mathematica. Zrovna tak není jeho standardní součástí podpora symbolických výpočtů [3].

To však neznamená, že je o tyto možnosti ochuzen. Obsahuje množství jednoduchých i složitějších matematických funkcí implementovaných ve formě vysoce efektivních a



Obrázek 1 – Prostředí programu MATLAB

robustních algoritmů, které jsou součástí jádra. Z těchto funkcí lze složit podle konkrétní aplikace libovolně složité další funkce. Skupiny takovýchto funkcí respektive funkcí hodících se k řešení určitého okruhu problémů se v MATLABu nazývají Toolboxy. Obvykle byly vytvořeny na univerzitách týmem seskupeným okolo významného odborníka v dané oblasti. Velkou výhodou, aspoň pro teoretické využití, je to, že dokumentace k toolboxům obsahuje stručný princip a odkazy na literaturu, kde jsou použité algoritmy podrobně rozepsány.

Popis důležitých částí programu MATLAB podle vyobrazení na obrázku 1:

1. Zobrazuje aktuální složku s uloženými projekty. Složku lze změnit v horní části tohoto okna za pomocí rolovacího okna, kde se nachází název s aktuální složkou.
2. Nejdůležitější část celého program. Příkazové okno, přes které se provádí za pomocí zadávání příkazů různé početní úlohy. Jaké příkazy je možno zadávat a jakou mají funkci, tak k tomu poslouží, jako ve všech programech tohoto typu, manuál (help).

3. Česky v překladu pracovní okno, kde se ukládají všechny proměnné, do kterých sám uživatel jej uložil za pomocí zadávání příkazů. Tím se dá snadno a přehledně překontrolovat uložené hodnoty.
4. Velmi praktická část nazvaná historie příkazů, kde se ukládají uživatelem zadané příkazy za sebou podle toho, v jakém pořadí byly zadávány. Ty jsou hierarchicky uloženy pod jeden velký celek a v hlavičce určený datum a čas, kdy byl celý program MATLAB spuštěn. Proto se může stát, že je v historii datum a čas bez příkazů, protože uživatel mohl využívat program pro jiný účel.

1.1.1 Kdy použít MATLAB

Záleží na konkrétních požadavcích a podmínkách konkrétního uživatele. V případě, že je zapotřebí robustní výpočet, zpracovávat rozsáhlé datové soubory, pracovat s velkými maticemi a vůbec v případech, kdy řešení problému se dá převést na vektorové či maticové operace. Vzhledem k možnostem programování je MATLAB s výhodou použitelný i v případech rozvětvených případně iteračních algoritmů řešení. Je běžným prostředkem používaným na většině univerzit technického zaměření na světě. Také to, že je k dispozici na všech běžných platformách může při výběru hrát roli [3].

1.1.2 Kdy není vhodné použití MATLABu

Použití MATLABu není vhodné v situaci, kdy je zapotřebí provádět opakovaně stále stejné, byť složité, výpočty nebo opakovaně zpracovávat velký objem dat. V situacích, kdy je rozhodující vlastní rychlost výpočtu a ne vývoj algoritmu výpočtu. V takových případech je vhodné zápis hotového algoritmu v určitém programovacím jazyce, který umožní vytvořit spustitelný kód a nepoužívat MATLAB, který je stále jen interpret [3].

1.2 SIMULINK

SIMULINK existuje jako nadstavba MATLABu. Jedním z jeho nejčastějších využití je řešení soustav (nelineárních) diferenciálních rovnic s grafickým zadáváním řešené soustavy připomínajícím zapojení na analogovém počítači. Umožňuje graficky sledovat průběhy veličin v libovolném místě zapojení. Praktické použití je např. pro simulaci dynamického chování sledovaného systému.

Rozšíření MATLABu, nadstavba, SIMULINK je k dispozici od verze 4. Využívá grafických možností hostitelské platformy Windows. Způsob zápisu modelu i celkový způsob práce je značně odlišný od práce s vlastním MATLABem, který je orientován na řádkové příkazy.

Tyto možnosti svádí k tomu, používat SIMULINK samostatně, nezávisle na znalosti MATLABu. I takovýto přístup je možný, ale vystačí pouze na nejjednodušší využití. Pro efektivní využití a při řešení složitějšího problému je znalost MATLABu prakticky nezbytná [3].

Na pozdější původ SIMULINKu ukazuje i jiný systém nápovědy. K dispozici je dobře vytvořená on-line nápověda, tlačítka help na jednotlivých blocích. Nápověda je ve formátu HTML a využívají se hypertextové odkazy. Určitou nevýhodou může být to, že je nutné mít instalovaný prohlížeč.

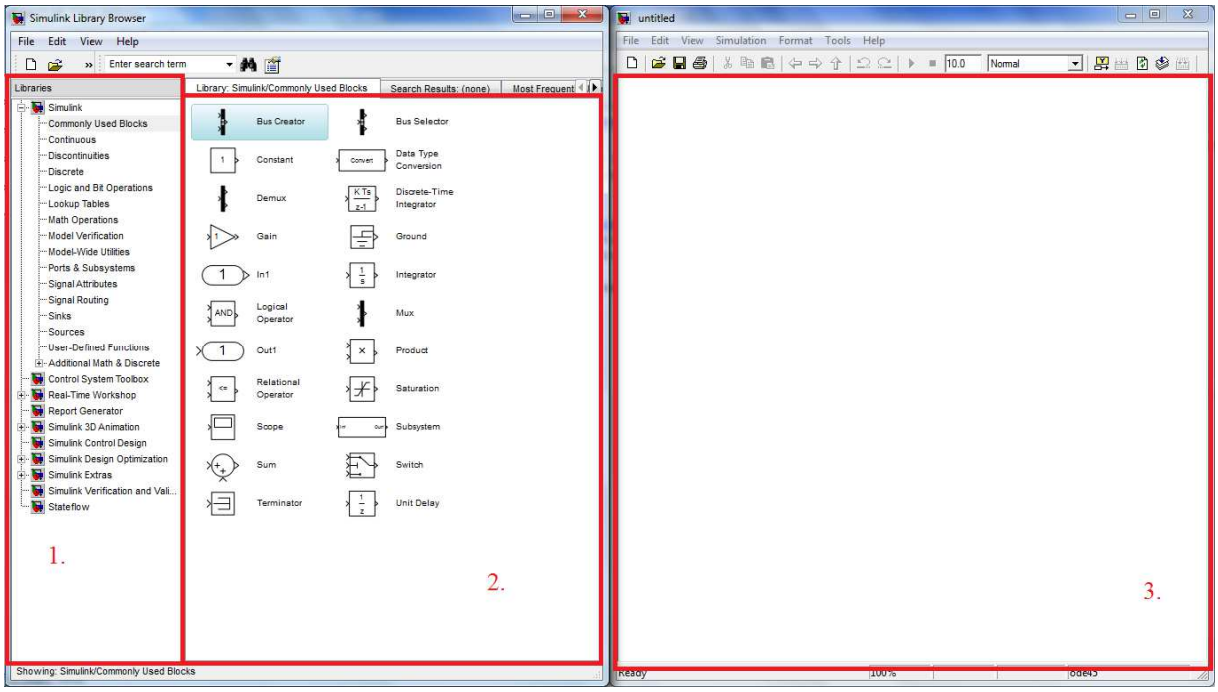
SIMULINK je určen na časové řešení (simulaci) chování dynamického systému pokud je známý jeho matematický popis. Lze s jeho pomocí určit časové průběhy výstupních veličin, a všech vnitřních, v závislosti na časovém průběhu veličin vstupních a počátečním stavu. Popis soustavy může být značně rozsáhlý a složitý.

Co nabízí je jednoduchý přístup k propracovaným metodám MATLABu pro časové řešení soustavy nelineárních diferenciálních rovnic, prostředky pro relativně snadný zápis problému a vizualizaci výsledků a automatické řešení mnoha problémů, které při řešení vznikají.

První co zaujme na pohled, je grafický způsob zápisu, v terminologii SIMULINKu nazývaný model. Z nabídky příslušné knihovny se myší přetahují výkonné bloky a pak se myší pospojují odpovídající vstupy a výstupy. Při tvorbě složitějších modelů se využívá možnosti část modelu sdružit do dalšího bloku a ty opět standardně kopírovat a spojovat. Jako vstupy modelu se používají nejčastěji signály z knihovních bloků generujících základní typy signálů, které lze libovolně poskládat. Lze také případně používat data ze souboru nebo z matic připravených v pracovním prostoru MATLABu [3].

Při bližším prozkoumání problematiky simulace chování dynamických systémů je zjištěno, že simulace není jen numerické řešení soustavy nelineárních diferenciálních rovnic, i když se jedná o nejdůležitější část. V konkrétním případě se musí řešit několik dalších problémů od výběru vhodné metody řešení pro daný problém a zahrnutí počátečních stavů, přes volbu kroku řešení (souvisí s požadovanou přesností a rychlostí výpočtů), problém nespojitých nelinearit (zahuštění bodů řešení v okolí nespojitosti), synchronizaci výpočtu na zadaný čas, problém algebraických smyček až po problém závislosti daných tabulkou. Právě automatické, i s manuální volbou, řešení některých problémů a podporu pro řešení dalších SIMULINK nabízí [3].

Kromě vlastního grafického rozhraní se standardními knihovnami, a v případném rozšíření knihovny z toolboxů, je SIMULINK podporován i příkazy a funkcemi dostupnými z prostředí MATLABu.



Obrázek 2 – Prostředí nadstavby SIMULINK

Popis důležitých částí nadstavby SIMULINK podle vyobrazení na obrázku 2:

1. Okno se stromově řazenými knihovnami. Název dané knihovny napovídá, jaké bloky bude nejspíše obsahovat.
2. Při výběru určité knihovny se zobrazí výkonné bloky, které mají svou funkci (logickou či matematickou). Pomocí jednoduchého stisknutí a přetažením bloku jej můžeme přidávat na pracovní plochu.
3. Pracovní plocha je nejdůležitější částí nadstavby, kde se tvoří celé obvody z použitých výkonných bloků. V okně je umožněno obvody simulovat v čase, který se nastavuje v horní části panelu. Za použití správných výkonných bloků je také umožněno sledování vstupních i výstupních hodnot.

1.3 Řešení diferenciálních rovnic Rungovou-Kuttovou metodou

Těchto metod lze užít nejen k získání počátečních hodnot, ale samozřejmě i celého řešení. Nevyžadují dodatečných počátečních hodnot a lze je snadno naprogramovat v jakémkoliv

programovacím jazyce, ale tyto přednosti nevyváží jejich nevýhody, totiž obtížné určení chyby a malou rychlost. Mají ovšem velkou cenu právě pro získání počátečních hodnot.

Pro vysvětlení a ukázku metody v tomto oddílu poslouží diferenciální rovnice prvního řádu

$$\frac{\mathrm{d}y}{\mathrm{d}x} = f(x,y), \quad y(x_0) = y_0 \tag{1}$$

Předpokládá se, že funkce $f(x,y)$ je definována a spojitá v pase $x_0 \leq x \geq b$, $-\infty < y < \infty$,

kde x_0 a b jsou konečná čísla [2].

Základem všech Rungových-Kuttových metod je vyjádření rozdílů mezi hodnotami y v bodech x_{n+1} a x_n ve tvaru:

$$y_{n+1} - y_n = \sum_{i=1}^{m} w_i k_i \,, \tag{2}$$

kde w_i – konstanta,

$$k_i = z_n f(x_n + \alpha_i z_n, y_n + \sum_{j=1}^{i-1} \beta_{ij} k_j), \tag{3}$$

kde $z_n = x_{n+1} - x_n$,

$\alpha_1 = 0$.

Užívá se označení z_n místo z, protože je možné měnit interval metody Runge-Kutta v každém stádiu, ale je tato změna zřídka účelná. Jsou-li předem dány w_i,α_i a β_{ij}, je pak rovnice (2) metodou pro řešení rovnice (1), která tím pádem nevyžaduje dodatečných počátečních podmínek.

Nejznámější a nejpoužívanější metodou Runge-Kutta je formule čtvrtého stupně. Odvození rovnic pro příslušné konstanty je značně pracné a zabíraly by značnou část bakalářské práce. Proto je odvození přeskočeno a uvedena konečná formulace

$$\begin{aligned}
&y_{n+1} - y_n = \frac{1}{6} z_n (k_1 + 2k_2 + 2k_3 + k_4), \\
&k_1 = f(x_n, y_n), \\
&k_2 = f(x_n + \frac{1}{2} z_n, y_n + \frac{1}{2} z_n k_1), \\
&k_3 = f(x_n + \frac{1}{2} z_n, y_n + \frac{1}{2} z_n k_2), \\
&k_4 = f(x_n + z_n, y_n + k_3).
\end{aligned} \tag{4}$$

Řeší-li se diferenciální rovnice pomocí uvedených vzorců (4), je vhodné všechny výpočty účelně organizovat. V případě této bakalářské práce předešlá informace je brána mezi hlavní priority, jelikož výpočty jsou organizovány programově do předem nadefinovaných polí, ale to až dále v praktické části. Postupné vypočítávání hodnot řešení $y_1,y_2,\ldots,y_n$ je řešeno za pomocí opakovaného používání vzorce (4). Z toho vyplývá, že největší díl práce zpravidla tvoří výpočet funkčních hodnot funkce *f*.

Pro porovnání metod Runge-Kutta s metodami predikce-korekce stejného řádu jsou uvedeny tyto skutečnosti:

- Rungovy-Kuttovy metody nevyžadují dodatečných počátečních hodnot, kde krok integrace z_n lze libovolně měnit a jejich využití pro samočinné počítače je velmi jednoduchá.
- S metodami predikce-korekce o stejném řádu mají stejnou přesnost, avšak často jsou metody Runge-Kutta přesnější. Je to dáno tím, že je obtížné stanovit jejich lokální chybu, proto se obyčejně volí krok dle svého uvážení.
- Dále metody potřebují tolik výpočtů funkčních hodnot $f(x,y)$, jaký je jejich řád. Metody predikce-korekce čtvrtého řádu potřebují pouze dva výpočty funkčních hodnot v jednom kroku. Při řešení rovnice (1) nejvíce času zabere výpočet funkčních hodnot $f(x,y)$. Z toho vyplývá, že metody predikce-korekce jsou při výpočtech dvakrát rychlejší než metody Runge-Kutta.

1.4 JAVA

Znamená v angličtině „kafé“, ale přesto se jedná o programovací jazyk pocházející od firmy Sun Microsystems, později koupené firmou Oracle. Jedná se o objektově orientovaný jazyk vycházející z C++, ke kterému má také syntakticky nejblíž – nepočítáme-li C#, který vznikl až po příchodu programovacího jazyku JAVA [8].

Oproti svému předchůdci Java neobsahuje některé konstrukce, které způsobovaly při programování největší potíže, a navíc přidává mnoho užitečných vlastností:

- Přidělování a uvolňování paměti je prováděno automaticky.
- Klasický problém z C/C++, ukazatele, je zde zcela odstraněn, neboť ty zde prostě nejsou (resp. jsou nahrazeny referencemi). Programátor je zde ušetřen notorické chyby zápisu pointerem mimo datovou oblast.

- Je implementován mechanismus vláken (threads) a lze tudíž spouštět více úloh v rámci jednoho programu. Bylo samozřejmě pamatováno i na jejich synchronizaci pomoci tzv. monitorů.
- Lze provádět reflexi neboli zjišťování informací o objektu.
- Je implementován mechanismus výjimek, takže veškeré runtime chyby je možné odchytit a zpracovat. Výjimky jsou samozřejmě objektové, takže lze zachytit i celou hierarchii výjimek v jednom hlídaném bloku.
- Značným ulehčením pro programátory je obsáhlost standardně dodávaných knihoven, se kterou se nemůže srovnávat asi žádný běžně používaný jazyk. K dispozici jsou knihovny pro tvorbu grafického uživatelského rozhraní (GUI), vstup/výstup, práci s textem, komunikaci s SQL databázemi, práci s komprimovanými soubory a mnoho dalších!
- Podporuje tvorbu dynamicky rozšiřitelných aplikaci.
- Klade značný důraz na bezpečnost. Díky tomu, že je překládaná do byte code, a implementovaným bezpečnostním mechanismům (podepisování kódu a přidělování práv pro různé akce).
- Lze zajistit, že program v jazyce JAVA, který si uživatel stáhne ze sítě, mu nezformátuje disk, nebude komunikovat s žádným jiným počítačem, než ze kterého pochází atd.
- JAVA je jazyk velice jednoduchý, přehledný a srozumitelný. Při osvojování základů dělá začátečník podstatně méně chyb než při studiu C/C++.

Největším přínosem jazyku JAVA je bezesporu plná přenositelnost programů na libovolnou platformu bez nutnosti jejich rekompilace. Programy se totiž nepřekládají do strojového kódu konkrétního procesoru, ale do nezávislé podoby, tzv. bytového kódu (byte code). Tento kód pak může být interpretován na jakémkoliv počítači nebo průmyslovém zařízení. Kompatibilita je tedy zajištěna na binární úrovni.

Hlavním praktickým nedostatkem jazyku JAVA je malá rychlost interpretovaných programů. To se týká zejména internetových prohlížečů, které navíc mají mnohdy problémy se stabilitou appletů, ovšem nikoliv vinou jazyku JAVA. Tento nedostatek však vyřešily JIT kompilátory. Jde vlastně o myšlenku dynamické (re-) kompilace kódu. To znamená, že dochází k překladu byte code v okamžiku, kdy se vykonává a tak se neukládá do externí cache.

Druhým problémem jsou zvýšené nároky na paměť, vznikající v důsledku automatické správy paměťových prostředků. Kapacita pamětí počítačů však neustále roste, a tudíž se jedná o poněkud méně závažný nedostatek.

1.5 GUI

Když JAVA vznikala, její tvůrci ušili grafickou část velice horkou jehlou. Původní návrh byl hodně nedokonalý, neumožňoval snadné a systematické využití. Proto se již s JDK 1.0 vytvořila u vývojářů jistá averze vůči javovské grafice. V dalších verzích však docházelo k neustálému zlepšování, takže se JAVA brzy stala vhodnou platformou pro tvorbu grafických aplikací.

1.5.1 AWT

První implementací grafiky byla knihovna zvaná AWT. Objevila se hned na začátku v JDK 1.0 a její nepříjemné a nesystematické API přetrvává v Javě dodnes. Něco se stále používá, ale jsou to spíš věci související s obecnou grafikou, a nikoli s GUI. V dalších verzích bylo sice API přepracováno a podstatně rozšířeno, ale jiné nevýhodné vlastnosti přetrvaly [9].

Filosofie AWT byla taková, že každá komponenta v Javě měla svůj nativní protějšek v systému. Byl to tedy podobný model, jako dnes používají některé GUI platformy v C/C++. I když se jednalo o abstraktní na platformě nezávislé rozhraní, přenositelnost byla problematická vzhledem k rozdílným vlastnostem nativní úrovně GUI [9].

1.5.2 Swing

Od verze 1.2 se nachází v jazyce JAVA nová grafická knihovna zvaná JFC, známá spíše pod názvem Swing, dále v textu bakalářské práce je již uváděno toto označení. Původní koncept byl zavržen, Swing byl vytvořen prakticky kompletně na zelené louce, i když v sobě obsahuje AWT API. Základní vlastnosti lze shrnout do těchto bodů:

- Kompletní implementace v Javě. GUI není napasováno na nativní komponenty, vše je zcela platformě nezávislé.
- Intenzivní využití dědičnosti a kompozice. Komponenty GUI využívají objektové vlastnosti Javy, složitější součásti jsou složeny či zděděny z jednodušších.
- Výrazné využití rozhraní. Různé operace v komponentách GUI (kreslení, editace apod.) se provádějí přes rozhraní. Standardní implementace lze nahrazovat vlastními a měnit tak chování komponent.
- Důsledné oddělení funkcionality od vzhledu GUI (look & feel). Témat vzhledu je několik k dispozici přímo ve standardní knihovně, další si lze vytvořit a použít.

- Oddělení dat od objektů GUI u složitějších komponent (strom, tabulka) pomocí tzv. modelů. To opět zlepšuje možnosti přizpůsobení a také opakovanou použitelnost komponent.

Uvedené vlastnosti, pro programátory velice příjemné, mají druhou stránku věci - relativně vysoké nároky na rychlost procesoru a hlavně na paměť. V reakci na to vznikly grafické knihovny, které se snaží tento problém odstraňovat. Nejvýznamnější z nich je SWT [9].

1.5.3 SWT

GUI knihovna s podobnými vlastnostmi, jako má Swing, ovšem s nativní implementací grafických komponent. Původní cíl vzniku, tedy snížení procesorové a paměťové náročnosti, příliš naplněn nebyl a srovnatelné aplikace postavené na SWT jsou zhruba stejně náročné jako ty co používají Swing [9].

1.5.4 Jak se kreslí

V grafickém rozhraní GUI se podstatě jedná o grafiku. Proto je zásadní, aby uživatel měl možnost, svůj program s grafickým výstupem, nakreslit. Ke kreslení se používá tzv. grafický kontext, což je implementace abstraktního kreslicího rozhraní.

Při kreslení se může využívat široká škála nástrojů, které jsou ve standardní knihovně k dispozici. Je dobré je používat do té míry, co nejvíc je to možné, protože ve vztahu ke grafickému kontextu většinou poskytují maximální možnou optimalizaci. Kreslení nebo spíš skládání grafického rozhraní je díky předpřipraveným komponentám snadnou záležitostí. Jednoduše pomocí tažení myši se komponenty skládají na grafický panel, kde každá komponenta, která je využitá se automaticky generuje do kódu. Těm pak může uživatel libovolně přidávat různé vlastnosti a funkčnosti jak kódově, tak je i možnost nastavení těchto parametrů v grafickém rozhraní [9].

1.6 Fyzikální zákony

1.6.1 Torricelliho zákon

Torricelliho zákon známý také pod Torricelliho větou se zabývá dynamikou kapalin. Konkrétně se to týká rychlosti vytékání kapalin v závislosti na výšce umístění otvorů v nádobě. Tím zjistil, že rychlost kapaliny vytékající z otvoru se rovná rychlosti volného pádu kapaliny z výšky, ve které se hladina kapaliny v nádobě v daný moment nachází. Přitom také objevil, že proud kapaliny vytékající z bočního otvoru nádoby má tvar paraboly. Těmito

poznatky dal Torricelli základy hydrodynamice a stanovil i pro tyto poznatky Torricelliho vzorec ve tvaru

$$v = \sqrt{2gh}\,, \tag{5}$$

kde v – rychlost kapaliny vytékající z otvoru,

g – tíhové zrychlení,

h – výška hladiny kapaliny v nádobě.

V bakalářské práci se počítá s tíhovým zrychlením v hodnotě g = 9,81 m.s^{-2}.

1.6.2 Potenciální energie

Potenciální energie nebo také polohová energie je druh energie, kterou má každé těleso nacházející se v potenciálovém poli určité síly. Sílu působící na těleso lze rozlišit na více druhů a to na gravitační potenciální energii, potenciální energii pružnosti, tlakovou potenciální energii, elektrostatickou potenciální energii. V této bakalářské práci se v podstatě jedná o tlakovou potenciální energii [12].

Tlaková potenciální energie je potenciální energie kapaliny nebo plynu, vznikající z tlaku, kterým kapalina tlačí na stěny nádoby. Může-li se stěna nádoby pohybovat (např. píst), pak kapalina posouváním pístu koná práci. Tlaková potenciální energie se mění na kinetickou energii pístu.

1.6.3 Kinetická energie

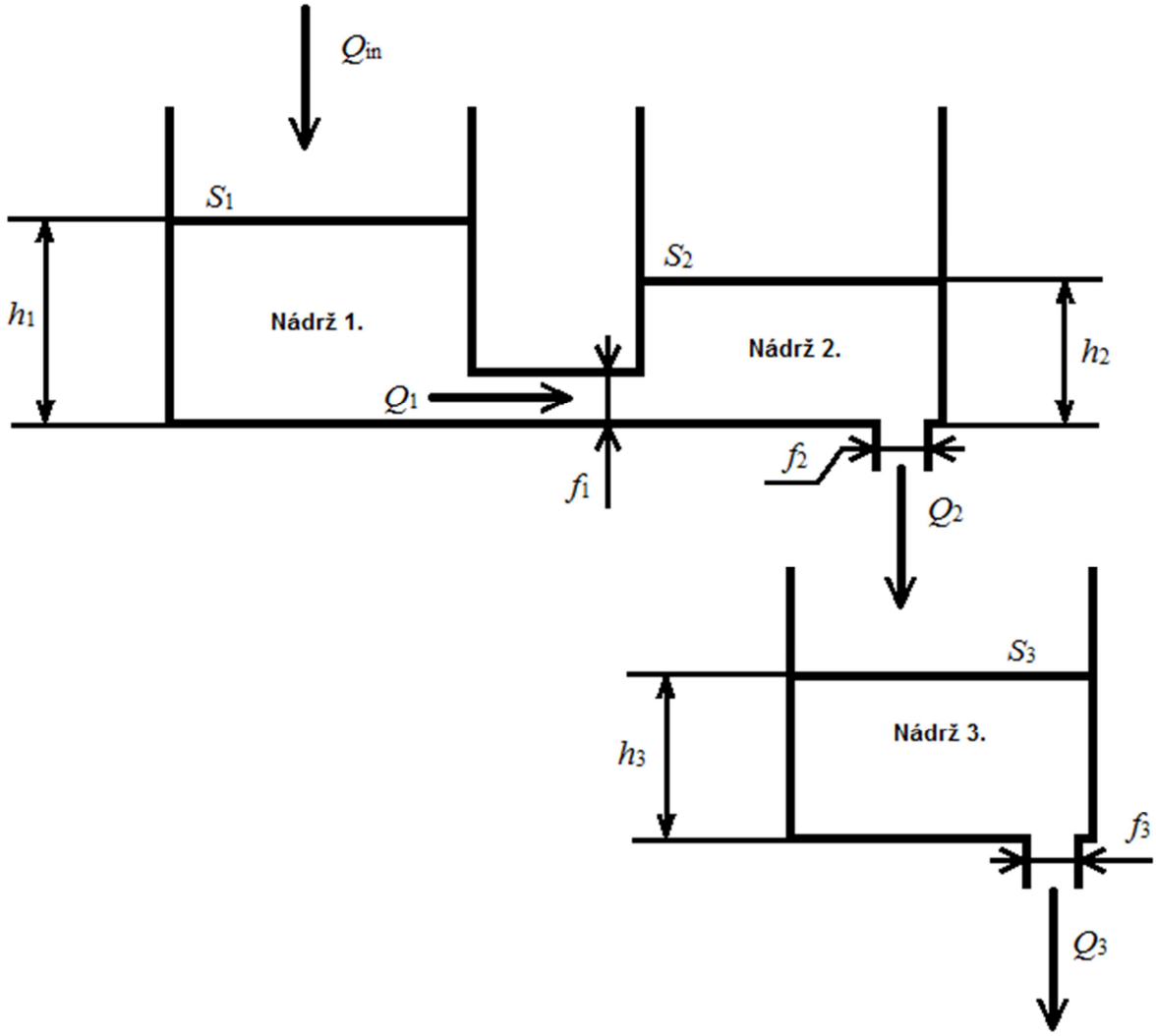
Jakmile se nějaký předmět pohybuje, znamená to, že má energii. Energie pohybujícího se tělesa se nazývá energie kinetická (též pohybová energie). Je-li těleso v klidu, má nulovou kinetickou energii. To znamená, že čím rychleji se těleso pohybuje, tím má větší kinetickou energii a naopak tím více energie musí vynaložit, než opět zabrzdí. Pokud se však stejnou rychlostí pohybuje těžší těleso, například automobil, má větší kinetickou energii díky své vyšší hmotnosti.

2 Praktická část

2.1 Návrh hydraulické soustavy

Úkolem bakalářské práce je navržení a simulace hydraulické soustavy v příslušném programu. Pro simulaci soustavy se používá nadstavba SIMULINK programu MATLAB. Před samotnou simulací je nutné navrhnout hydraulickou soustavu, což záleží na vlastním

uvážení, tj. kolik nádrží se použije a jak budou umístěny. Pro řešení bakalářské práce jsou vybrány tři nádrže válcovitého tvaru, které jsou napouštěny vodou. Umístění nádrží je zvoleno tak, že dvě nádrže se nacházejí a jsou propojeny vedle sebe, avšak třetí nádrž je pod nádrží označenou jako číslo dvě, ze které je vyveden otvor jako výpusť. Nádrže umístěné vedle sebe se nazývají systémem se zpětným ovlivňováním a nádrže umístěné pod sebou jsou nazývány systémem bez zpětného ovlivňování. Přítok vody se nachází nad jednou z nádrží, která má označení hodnotou jedna. Pro lepší přehlednost je navržená hydraulická soustava vyobrazena níže na obrázku 3.



Obrázek 3 – Grafické vyobrazení navržené hydraulické soustavy

Popis jednotlivých značek na obrázku 3:

- Q_{in} – hlavní přítok vody
- Q_1, Q_2, Q_3 – průtoky jednotlivých nádrží
- h_1, h_2, h_3 – výška hladiny vody pro příslušnou nádrž

- f_1, f_2, f_3 – průřez průtoků nádrží válcovitého tvaru
- S_1, S_2, S_3 – plocha hladin nádrží

Návrh nádrže vyobrazený výše na obrázku 3 je sestaven tak, aby se využívalo obou kombinací systému. Tím je myšleno, jak je popsáno výše, systému se zpětnovazebním ovlivňováním a bez zpětného ovlivňování. Účelem je tak ukázat vliv rozložení na výšku hladin jednotlivých nádrží. Avšak pouhým návrhem systému tato bakalářská práce nekončí. Je nutné tuto hydraulickou soustavu také popsat pomocí tzv. bilančních rovnic.

2.2 Popis systému pomocí bilančních rovnic

Jak bylo uvedeno v předešlém bloku, je za potřebí pro další práci s hydraulickou soustavou popsat tento systém pomocí bilančních rovnic. Taková rovnice je však konečný popis systému, a proto je dobré si nejdříve uvědomit různé aspekty, které ovlivňují celou hydraulickou soustavu. Tomu předchází matematicko-fyzikální analýza, která se zakládá v podstatě na aplikaci předem známých fyzikálních vzorců a zákonů a ty jsou sestaveny pomocí matematických vztahů a výpočtů. Systém je ovlivňován především fyzikálními zákony a to konkrétně zákonem o zachování energie. Ta je schopna se měnit z jednoho druhu energie na druhý druh energie přičemž nezaniká, ale zachovává se. V tomto případě dochází k přeměně potenciální energie na energii kinetickou, které jsou popsány výše v teoretické části.

Pro popis rovnicemi je vybrána nádrž č. 1, která je vyobrazena v systému na obrázku 3. U nádrží napouštěné vodou opatřené výtokem je důležité si uvědomit, že podstatě v nich dochází k přeměně potenciální energie na kinetickou energii. Tuto přeměnu se tak může popsat rovnicí

$$h\rho g = \frac{1}{2}\rho v^2 \ , \tag{6}$$

kde ρ – hustota kapaliny.

V tomto případě, jak již bylo zmíněno, se jedná o vodu. Průtok kapaliny lze vyjádřit jako

$$Q = v.f \ . \tag{7}$$

Tím vztah (7) je možné dosadit do rovnice (6) a získá se vztah

$$h\rho g = \frac{1}{2}\rho\frac{Q^2}{f^2},$$

$$\frac{Q^2}{f^2} = 2gh, \tag{8}$$

$$\underline{\underline{Q = f\sqrt{2gh}}}.$$

Obecná rovnice (8) je v podstatě Torricelliho vztah. Ta však není výslednou rovnicí průtoku, protože umístění nádrže č. 1 a č. 2 odpovídá systému se zpětným ovlivňováním. Proto obecnou rovnici (8) je nutné upravit pro tento systém a to následnou úpravou

$$Q_1 = f_1\sqrt{2g\left(h_1 - h_2\right)}, \tag{9}$$

kde h_1 – výška hladiny v nádrži č. 1,

h_2 – výška hladiny v nádrži č. 2,

Q_1 – průtok kapaliny z nádrže č. 1.

Dále je nutné si uvědomit, že tato nádrž se napouští a zároveň za pomocí průtoku vypouští. Tudíž se mění i objem vody v nádrži s postupem času. Pro tento případ se použije rovnice

$$Q_{\text{in}} = Q_1 + \frac{\mathrm{d}V}{\mathrm{d}t}, \tag{10}$$

kde $\frac{\mathrm{d}V}{\mathrm{d}t}$ – derivace objemu kapaliny v nádrži podle derivace času.

Avšak v této práci se sleduje výška hladin kapaliny v nádržích, a proto se musí upravit rovnice (10). Pro tento účel slouží vztah

$$\frac{\mathrm{d}V}{\mathrm{d}t} = \frac{\mathrm{d}(Sh)}{\mathrm{d}t} = \underline{\underline{S\frac{\mathrm{d}h}{\mathrm{d}t}}}. \tag{11}$$

Dále je tedy možné dosazení rovnice (11) do rovnice (10) a tím se získá následující rovnice

$$Q_{\text{in}} = Q_1 + S_1\frac{\mathrm{d}h_1}{\mathrm{d}t}. \tag{12}$$

Výsledná rovnice se tak nazývá onou bilanční rovnicí. Avšak rovnice popisující systém není pouze bilanční rovnice (12), ale také rovnice (9).

Pro praktičnost a logičnost je důležité si uvědomit, že uživatelé těžko budou hned znát plochy a průřezy průtoků nádrží, ale jistě budou vědět jejich průměry. Již na začátku v předešlém

bloku je uvedeno, že se jedná o nádrže a průtoky válcovitého tvaru. Válec jakožto třírozměrné těleso tvoří podstavy ve tvaru kruhu. Tedy je nutné vypočítat plochu kruhu

$$S = f = \frac{\pi d^2}{4}\ , \tag{13}$$

kde d – průměr nádrže/průtoku.

Jedná se o obecný vzorec, který je používán pro výpočet obou ploch, jak nádrže, tak i průtoku. Na základě vzorců vypočtených výše lze již sestavit bilanční rovnice i pro zbylé nádrže. Jelikož nádrž č. 2 a č. 3 jsou umístěny pod sebou, tak stačí pro popsání rovnice obecný vztah (8). Avšak u bilanční rovnice (12) je nutné pozměnit indexování u proměnných a uvědomit si co je vlastně vstupním a výstupním parametrem. Na základě poznatků pro nádrž č. 2 vyhází obě rovnice ve tvaru

$$\begin{aligned} Q_1 &= Q_2 + S_2 \frac{\mathrm{d}h_2}{\mathrm{d}t}, \\ Q_2 &= f_2 \sqrt{2gh_2}. \end{aligned} \tag{14}$$

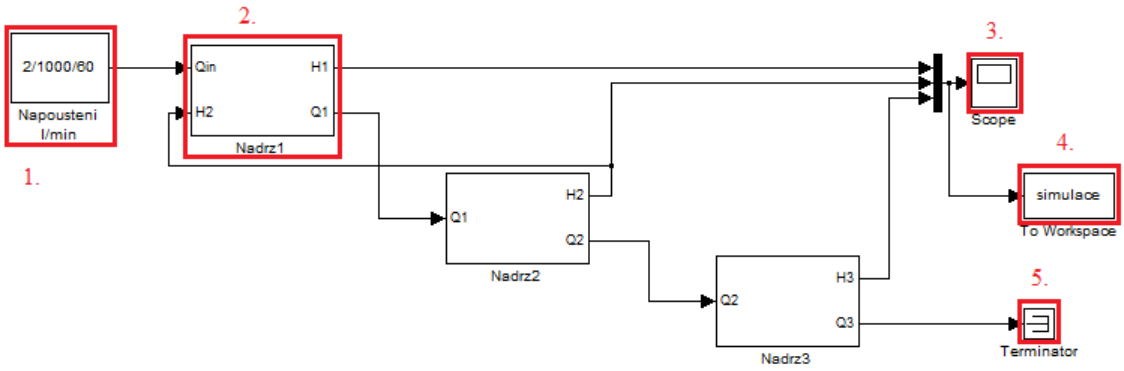
Rovnice (14) pro nádrž č. 3 jsou podstatě stejná, jenom se změní indexování, avšak pro přehlednost se uvádí rovnice ve tvaru

$$\begin{aligned} Q_2 &= Q_3 + S_3 \frac{\mathrm{d}h_3}{\mathrm{d}t}, \\ Q_3 &= f_3 \sqrt{2gh_3}. \end{aligned} \tag{15}$$

Tím jsou všechny nádrže popsány pomocí bilančních rovnic a může se dále s nimi pracovat. V bakalářské práci se využívají tyto rovnice konkrétně pro simulaci hydraulické soustavy v jazyce JAVA.

2.3 Simulace hydraulické soustavy

Nyní přichází na řadu ta praktičtější část a to simulace hydraulické soustavy na základě poznatků z předešlého bloku. Pro tento účel se využívá nadstavby SIMULINK. Díky tomu, že tato nadstavba umí pracovat s diferenciálními rovnicemi, je v podstatě implementace zjištěných rovnic snadnou záležitostí. Na základě toho se dají všechny rovnice pro nádrže sestavit. To, že je možné vytvářet tzv. subsystémy, usnadňuje práci a nastavování hodnot zvlášť pro každou nádrž. Práce s těmito bloky a subsystémy se popisují v bakalářské práci níže v jednotlivých obrázcích.



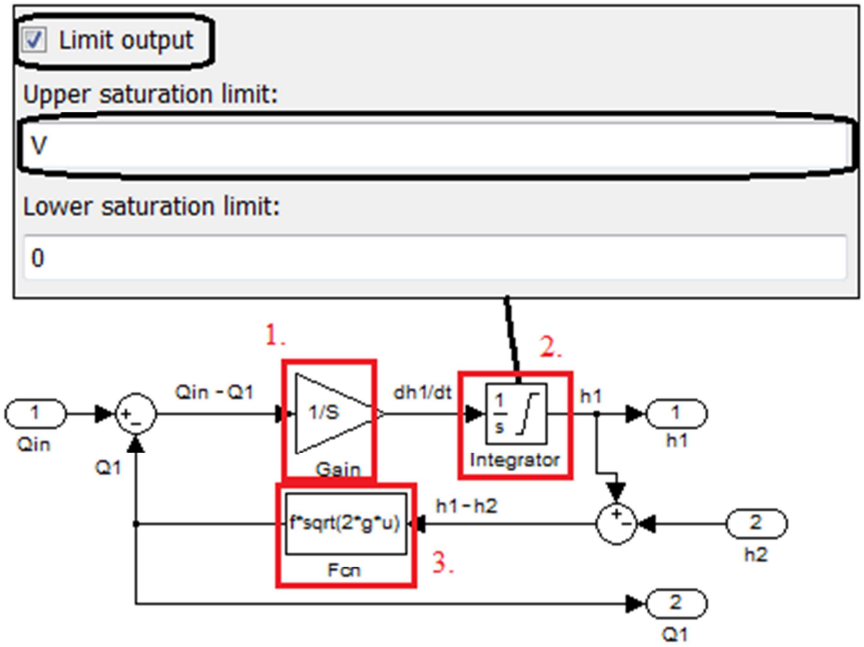
Obrázek 4 – Sestavená hydraulická soustava v SIMULINKu

Popis označených bloků na obrázku 4:

1. Blok označovaný jako konstanta. Hodnota zapsána v tomto bloku je při simulaci stála a nemění se. Proto se blok používá pro nastavení napouštění celé soustavy. Pro tento případ je nastavena hodnota napouštění na 2 $l.min^{-1}$.
2. Subsystém konkrétně pro nádrž č. 1, kde je tento subsystém popsán rovnicemi. V tomto případě rovnicí (8) a bilanční rovnicí (11). Zbylé další dva bloky jsou obdobně sestaveny pro konkrétní nádrž.
3. Pouze zobrazovací blok pro grafy. Jedná se o jednoduchý *x*, *y* graf. Blok zobrazuje výstupní proměnnou, v tomto případě výšky hladin jednotlivých nádrží v závislosti na čase.
4. Blok určený pro export nasimulovaných dat do Workspace v MATLABu. V simulaci je tento blok velmi příhodným společníkem, protože je možné po exportu do MATLABu dále pracovat s nasimulovanými hodnotami. V této práci však sloužila tato možnost exportu na kontrolu hodnot s hodnotami nasimulované v GUI.
5. Jakmile je některý výstup volný a nemá se kam připojit, tak k tomu slouží právě tento blok, který ošetří nepřipojený výstup. Tím se předejde případným chybám, které by mohly nastat.

Na obrázku 4 je pouze vyobrazeno obecné zapojení, ale podstatě není vidět co je uvnitř subsystémů. Ty však jsou tou hlavní částí celé simulace. Pro názornou ukázku je vybrán subsystém nádrže č. 1. Pro objasnění zapojení bloků na níže zobrazeném obrázku 5, bude nutné do bilanční rovnice (12) dosadit rovnici průtoku (9) a upravit do následující podoby

$$
\begin{aligned}
S\frac{\mathrm{d}h_1}{\mathrm{d}t} &= Q_{\mathrm{in}} - Q_1,\\
\frac{\mathrm{d}h_1}{\mathrm{d}t} &= \frac{Q_{\mathrm{in}} - Q_1}{S},\\
\underline{\underline{\frac{\mathrm{d}h_1}{\mathrm{d}t} = \frac{Q_{\mathrm{in}} - f_1\sqrt{2g(h_1 - h_2)}}{S_1}}}&.
\end{aligned}
\tag{16}
$$



Obrázek 5 – Vnitřní zapojení subsystému

Na obrázku 5 je patrné, že Q_{in}, h_2 jsou vstupními parametry a Q_1, h_1 jsou výstupními parametry. Pro lepší orientaci se na obrázku 5 zobrazují popisky spojů, aby bylo patrné, čí hodnoty se ve spoji předávají. Zapojení odpovídá rovnici (16).

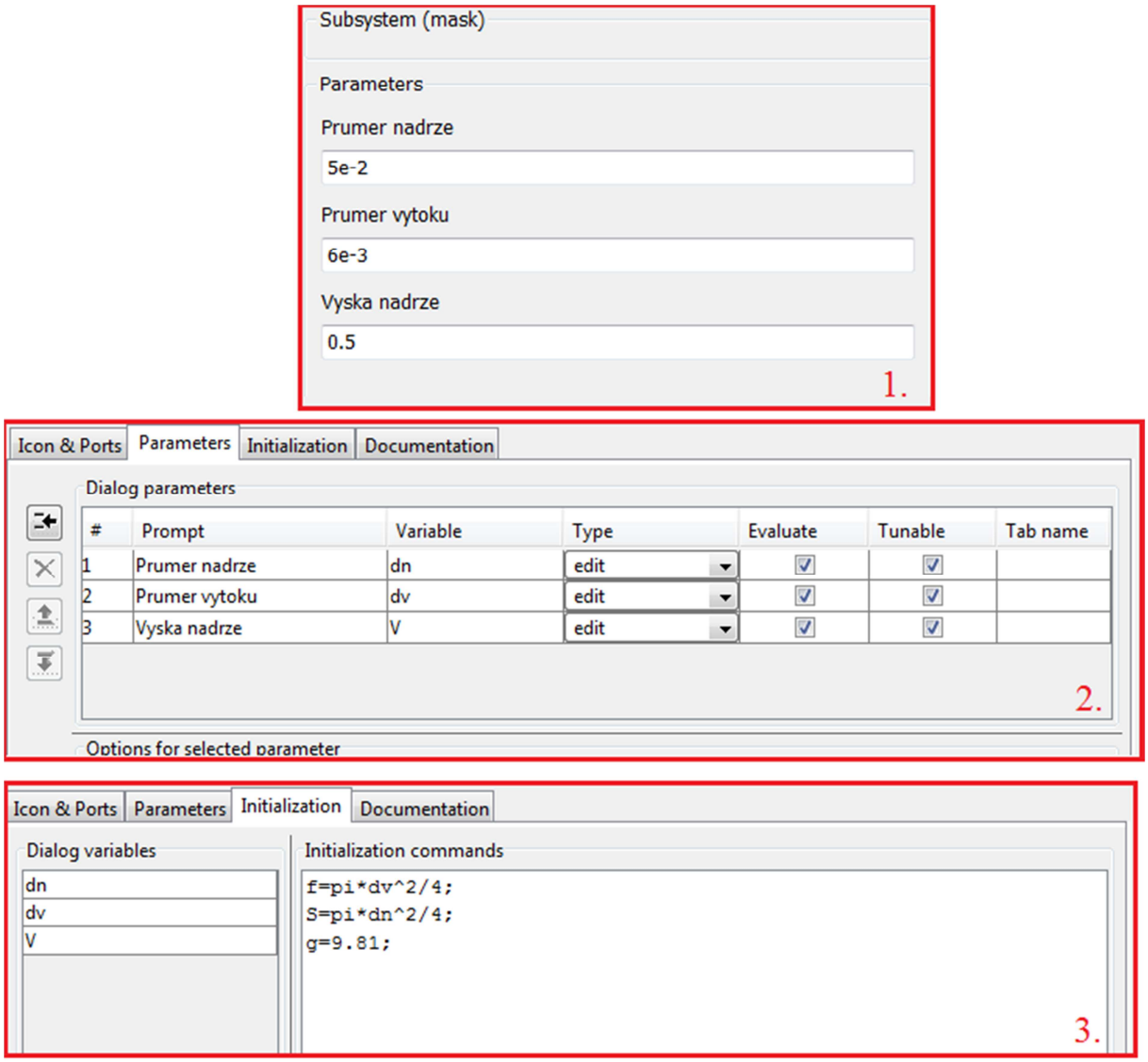
Popis vyznačených bloků na obrázku 5:

1. Z anglického překladu se jedná v podstatě o „zisk“. Tento příhodný název má právě tu funkčnost, kterým je tento blok pojmenován. Hodnota přivedená na vstup je násobena hodnotou bloku. V tomto případě je vstup násoben konstantou $1/S$.
2. Jedná se o integrační blok. V otevřeném okně nad schématem je patrné, že jde o omezenou integraci, protože je zaškrtnut checkbox „Limit output“. To v překladu znamená limitní výstup nebo také omezený výstup. V tomto případě je v políčku „Upper saturation limit“ nastaveno V a v políčku „Lower saturation limit“ je vyplněna nula. V doslovném překladu políček se jedná o horní a dolní mez nasycení, kde je tedy jasné, že při nastavení horní meze značkou V se bude jednat o výšku nádrže. Protože ta nám

omezuje, jaká bude maximálně možná výška hladiny v nádrži. Kdyby nebyla nastavena žádná mez, tak by podstatě byly nádrže nekonečně vysoké a nikdy by nedošlo k přetečení.

3. Funkční blok určený pro výpočet funkcí. V tomto případě se jedná o funkci nebo spíše o rovnici (9). Avšak ve vzorci uvedeném v bloku je násobeno proměnnou *u*. Ta představuje vstupní proměnou, protože v nadstavbě SIMULINK se obecně označují vstupy proměnnou *u* a výstupy proměnnou *y*. Vstupním parametrem tak je vzorec h_1-h_2.

Po sestavení blokového schématu pro diferenciální rovnici je zahrnuto do jednoho funkčního bloku, tedy subsystému. Pro subsystém lze nastavit ovládací panel pro zadávání hodnot k výpočtu. Takový panel je zobrazen níže na obrázku 6, kde je názorně zobrazeno i konkrétní nastavení pro nádrž č. 1. U nádrží č. 2 a č. 3 je změněn pouze průměr výtoku a to na 4 mm.

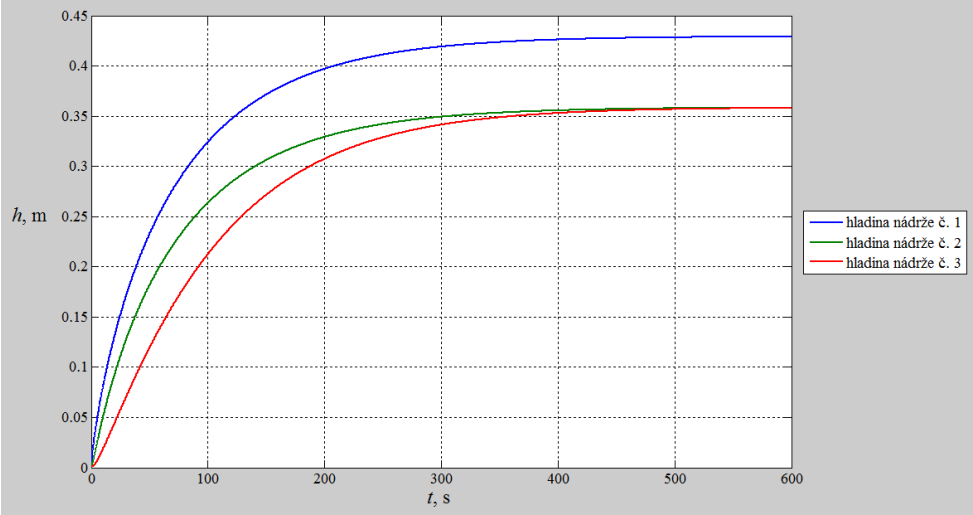


Obrázek 6 – Nastavení masky subsystému pro nádrž č. 1

Vysvětlení jednotlivých panelů na obrázku 6:

1. Jak hlavička panelu napovídá, tak se jedná o panel subsystému s nastavením parametrů. Jednotlivé parametry jsou pojmenovány dle toho, jakou proměnou pro subsystém představují. Konkrétně je možno nastavit průměr nádrže, průměr výtoku a výšku nádrže. V SIMULINKu lze matematicky zapsat desetinné jednotky pomocí zápisu např. 6e-3, což znamená ve skutečnosti hodnotu 6.10^{-3}. Bere se tedy i na vědomí, že základní jednotkou délky je právě metr. Tedy dle uvedeného příkladu je hodnota nastavena na 6 mm.
2. Panel pro vytváření parametrů, zmíněné v předešlém bodě. Parametry se přidávají pomocí tlačítka, které je umístěno vlevo od dialogového okna s parametry. Jak je vidno, názvy parametrů souhlasí s panelem popsaným v předešlém bodě. Sloupeček „Variable“ slouží pro nastavení proměnných, se kterými je možno pracovat přímo v subsystému. Jako příklad lze uvést výšku nádrže, kde je nastavena proměnná *V*, která byla popsána na obrázku 5 v bodě 2.
3. Podle uvedeného názvu v záložce se doslova jedná o inicializaci – nastavení. Tedy je možné pomocí příkazů používaných v MATLABu inicializovat nebo počítat s proměnnými. Z předešlého bodu jsou proměnné nastaveny právě pro výpočet ploch dle vzorce (13). Konkrétně jsou využívány proměnné d*n* a d*v*, průměr nádrže a průměr výtoku.

S nastaveními uvedenými v jednotlivých obrázcích je hydraulická soustava připravena k simulaci. Ta se provádí neustálými opakovanými změnami hodnot parametrů, aby došlo k ustálení hladin v jednotlivých nádržích. Výsledným nastavením hodnot parametrů se zajistí, že hladiny nádrží jsou z větší části naplněny, aby se lépe vystihla funkčnost celé soustavy. Výsledek se však musí někam zobrazit. K tomu nám poslouží, jak bylo popsáno pro obrázek 4 v bodě 1., zobrazovací blok v SIMULINKu tzv. „Scope“.



Obrázek 7 – Graf výšky hladin nádrží v závislosti na čase

Pro ukázku v této bakalářské práci je vyobrazen graf na obrázku 7 z programu MATLAB. U zobrazovacího bloku v SIMULINKu není možnost pojmenovávat jednotlivé osy a přidávat legendu. Proto jsou data vytažena do programu MATLAB pomocí bloku popsaném z obrázku 4 v bodě 4. Nakonec příkazy zadanými do příkazového okna se vykreslil graf zobrazený na obrázku 7.

Příkazy zadané v programu MATLAB v příkazovém okně:

```
>> t=simulace.time;
>> h1=simulace.signals.values(:,1);
>> h2=simulace.signals.values(:,2);
>> h3=simulace.signals.values(:,3);
>> plot (t,h1,t,h2,t,h3)
```

2.4 Použití metody Runge-Kutta na diferenciální rovnice nádrží

Již v teoretické části bakalářské práce je vysvětlena numerická metoda užívající se k řešení diferenciálních rovnic. Protože jazyk JAVA neumí pracovat s diferenciálními rovnicemi, tak z tohoto důvodu se musí použít metoda Runge-Kutta. V teoretické části je uvedeno obecné řešení této metody. Konkrétně se jedná o uvedenou formulaci (4). Ta se aplikuje na jednotlivé diferenciální rovnice nádrží. Pro nádrž č. 1 je již diferenciální rovnice (16) výslednou rovnicí, není ji potřeba nikterak upravovat. Pro zbylé dvě nádrže je však nutno, pro další práci s nimi, sestavit jejich vlastní diferenciální rovnice. Na základě výsledných rovnic (14) pro nádrž č. 2 a rovnice (9) se provede úprava

$$\begin{aligned} S\frac{\mathrm{d}h_2}{\mathrm{d}t} &= Q_1 - Q_2, \\ \frac{\mathrm{d}h_2}{\mathrm{d}t} &= \frac{Q_1 - Q_2}{S}, \\ \frac{\mathrm{d}h_2}{\mathrm{d}t} &= \frac{f_1\sqrt{2g(h_1 - h_2)} - f_2\sqrt{2gh_2}}{S_2}. \end{aligned} \tag{17}$$

Pro nádrž č. 3 z výsledných rovnic (15) se provádí, jako v předešlém kroku, stejná úprava

$$
\begin{aligned}
S\frac{\mathrm{d}h_3}{\mathrm{d}t} &= Q_2 - Q_3,\\
\frac{\mathrm{d}h_3}{\mathrm{d}t} &= \frac{Q_2 - Q_3}{S},\\
\frac{\mathrm{d}h_3}{\mathrm{d}t} &= \frac{f_2\sqrt{2gh_2} - f_3\sqrt{2gh_3}}{S_3}.
\end{aligned}
\tag{18}
$$

Výsledné diferenciální rovnice se použijí právě pro úpravu metodou Runge-Kutta. Důležité je si uvědomit, že v podstatě všechny tři diferenciální rovnice jsou hladiny nádrží závislé na čase. Představují tak funkci $f(x,y)$ tedy po dodržení závislosti a označení se dostane funkce $f(t,h)$.

Nejdříve se tak aplikuje numerická metoda na diferenciální rovnici (16) nádrže č. 1, kde po úpravě a pomocí poznatků z teoretické části o metodě je výsledkem formulace

$$
\begin{aligned}
k_1 &= \frac{Q_{\mathrm{in}} - f_1\sqrt{2g(h_1 - h_2)}}{S_1},\\
k_2 &= \frac{Q_{\mathrm{in}} - f_1\sqrt{2g\left[\left(h_1 + z\cdot\frac{k_1}{2}\right) - h_2\right]}}{S_1},\\
k_3 &= \frac{Q_{\mathrm{in}} - f_1\sqrt{2g\left[\left(h_1 + z\cdot\frac{k_2}{2}\right) - h_2\right]}}{S_1},\\
k_4 &= \frac{Q_{\mathrm{in}} - f_1\sqrt{2g[(h_1 + k_3) - h_2]}}{S_1},\\
h^1_{\mathrm{n+1}} &= h^1_{\mathrm{n}} + \frac{1}{6}z(k_1 + 2k_2 + 2k_3 + k_4),
\end{aligned}
\tag{19}
$$

kde z – zvolený krok,

$h^1{}_{\mathrm{n+1}}$ – výška hladiny v nádrži v následujícím kroku,

$h^1{}_{\mathrm{n}}$ – výška hladiny v nádrži v tomto kroku.

V dalším pořadí se pro nádrž č. 2 provede, jako v předešlém kroku, stejná úprava

$$
\begin{aligned}
k_1 &= \frac{f_1\sqrt{2g(h_1-h_2)}-f_2\sqrt{2gh_2}}{S_2},\\
k_2 &= \frac{f_1\sqrt{2g\left[h_1-\left(h_2+z\cdot\frac{k_1}{2}\right)\right]}-f_2\sqrt{2g\left(h_2+z\cdot\frac{k_1}{2}\right)}}{S_2},\\
k_3 &= \frac{f_1\sqrt{2g\left[h_1-\left(h_2+z\cdot\frac{k_2}{2}\right)\right]}-f_2\sqrt{2g\left(h_2+z\cdot\frac{k_2}{2}\right)}}{S_2},\\
k_4 &= \frac{f_1\sqrt{2g[h_1-(h_2+k_3)]}-f_2\sqrt{2g(h_2+k_3)}}{S_2},\\
h_{n+1}^2 &= h_n^2+\frac{1}{6}z(k_1+2k_2+2k_3+k_4).
\end{aligned}
\tag{20}
$$

A nakonec úprava pro nádrž č. 3 vypadá následovně

$$
\begin{aligned}
k_1 &= \frac{f_2\sqrt{2gh_2}-f_3\sqrt{2gh_3}}{S_3},\\
k_2 &= \frac{f_2\sqrt{2gh_2}-f_3\sqrt{2g\left(h_3+z\cdot\frac{k_1}{2}\right)}}{S_3},\\
k_3 &= \frac{f_2\sqrt{2gh_2}-f_3\sqrt{2g\left(h_3+z\cdot\frac{k_2}{2}\right)}}{S_3},\\
k_4 &= \frac{f_2\sqrt{2gh_2}-f_3\sqrt{2g(h_3+k_3)}}{S_3},\\
h_{n+1}^3 &= h_n^3+\frac{1}{6}z(k_1+2k_2+2k_3+k_4).
\end{aligned}
\tag{21}
$$

Všechny provedené úpravy jsou tak výsledkem použití numerické metody Runge-Kutta. Takto upravené rovnice se použijí pro práci v jazyce JAVA.

2.5 Hydraulická soustava v jazyce JAVA pomocí GUI

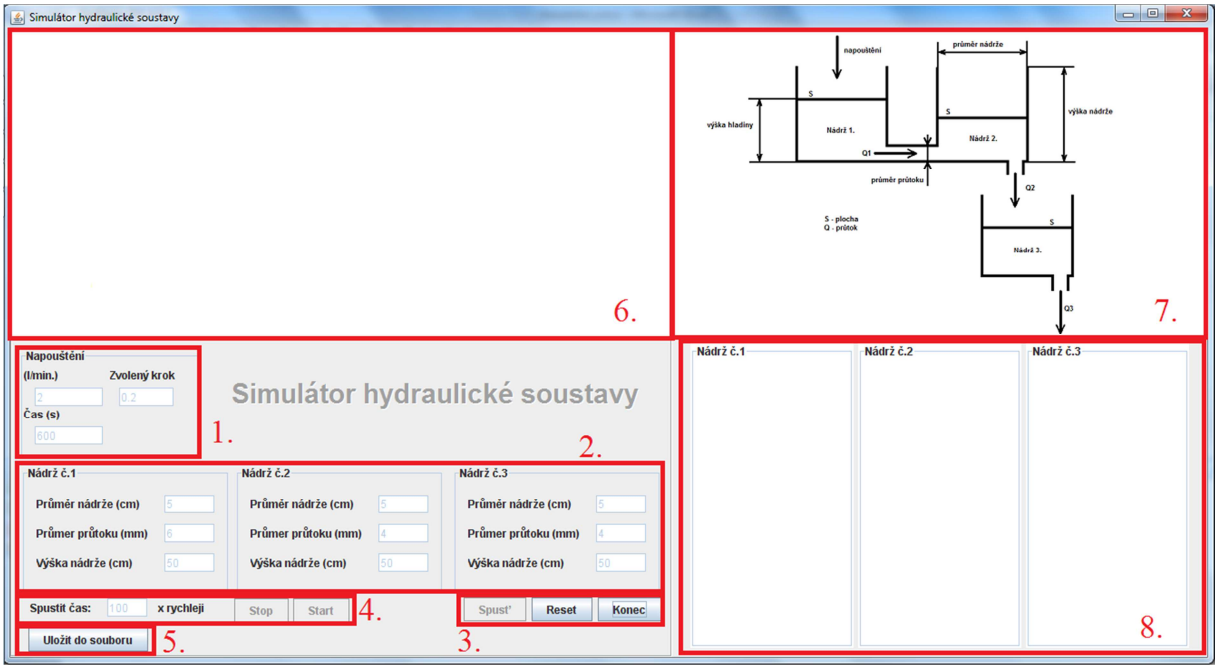
2.5.1 Aplikace Rungových-Kuttových metod

V části, kde je popsáno použití Rungových-Kuttových metod na diferenciální rovnice, jsou stanoveny rovnice pro jednotlivé nádrže, které se využívají právě pro práci v jazyce JAVA. Ukázkou tak v bakalářské práci je nejdůležitější část kódu v příloze A, kde právě se počítá s upravenými rovnicemi pomocí metody Runge-Kutta. Zápis metody v kódu se tolik neliší od

matematického zápisu rovnic (19), (20) a (21). V kódu je nutné využít některých předpřipravených matematických funkcí, které nelze běžně zapsat a to například odmocnina. Avšak pokud se nachází pod odmocninou hodnota nula, tak v jazyce JAVA tato matematická funkce vrací hodnotu jako neznámou. To však je ošetřené pomocí příkazů if a else, kde v případě výskytu nuly pod odmocninou se „natvrdo“ nastaví místo odmocniny nula, což je výsledkem běžné kalkulačky. Celý výpočet je organizován v jedné metodě, aby byl kód více přehledný a při výskytu chyb, byly lehce dohledatelné. Z toho důvodu se využívá příkaz switch, kterým se velmi přehledně dají rozdělit jednotlivé rovnice pro nádrže, jak je patrné v příloze A. Pak se jednoduše v kódu metoda použije a na základě zadané hodnoty se pomocí příkazu switch vybere konkrétní rovnice. Je využita také univerzální metoda pro výpočet povrchu nádrží a výtoků, aby zbytečně se nekomplikovali rovnice a byly tak přehledné k lepšímu pochopení.

2.5.2 Simulace pomocí grafického rozhraní GUI

Stěžejní část zadání je vytvořit program v jazyce JAVA pomocí grafického rozhraní GUI. Pomocí jednoduché implementace panelů se v GUI poskládá a realizuje aplikace, která má svoji funkčnost a po grafické stránce vypadá velmi přehledně. Jednotlivé komponenty jsou viditelné níže na obrázku.



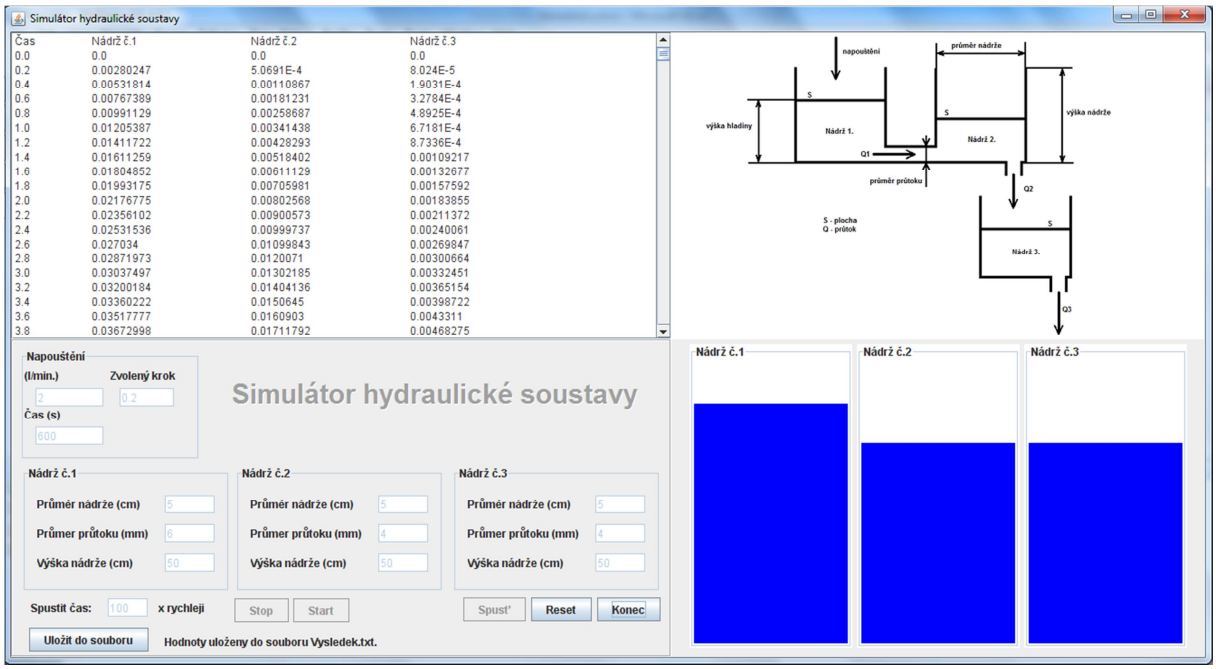
Obrázek 8 – Grafické vyobrazení aplikace

Popis bodů na obrázku 8, vysvětlení jejich vlastností a funkčnosti:

1. Jak sám název bloku napovídá, jedná se o napouštění celé hydraulické soustavy. Uživatelé mají možnost nastavit, kolik kapaliny bude do nádrží připouštěno, jakou dobu má simulace trvat a zvolit si jaké bude krokování času, při kterém budou vypisovány hodnoty výšky nádrží. Všechny parametry mají nadepsáno, v jakých jednotkách je zadaná hodnota vyjádřena. Nastavené hodnoty v obrázku 8 jsou reálné hodnoty nasimulované v nadstavbě SIMULINK. Samotná aplikace však má hodnoty v základu nastavené na nulu.
2. Každá nádrž má vlastní nastavení parametrů, kde je vše přehledně rozděleno do samostatných bloků opatřených popiskami. Opět jako v předchozím bodě je uživatel informován, v jakých jednotkách je zadaná hodnota vyjádřena. Jedná se o nastavení parametrů jako je výška nádrže, průměr nádrže a průřez průtoku. Všechny tyto parametry a schéma zapojení jsou graficky zobrazeny a popsány na obrázku 8 v označeném poli č. 7. Opět jsou v polích nastaveny reálně nasimulované hodnoty, které pouze slouží pro názornou ukázku. Hodnoty, jako v předešlém bodě, jsou nastaveny na hodnotu nula.
3. Tlačítka určená pro ovládání celé simulace. Tím je míněno spouštění simulace, resetování nasimulovaných hodnot a možnost ukončení celé aplikace. Funkčnost tlačítka „Spusť“ je taková, že po stisku začne celá simulace soustavy. Tím se tedy provádí výpočet dle bilančních rovnic popsaných výše a dále je nastaveno chování tlačítek tak, že se zviditelní např. tlačítka „Reset“, „Konec“ a všechny textová políčka v ovládacím panelu aplikace se zneviditelní, aby uživatel neměnil za běhu hodnoty a neovlivňoval tak celou simulaci. K tomu poslouží tlačítko „Reset“, které podstatě vymaže všechna nasimulované hodnoty a zviditelní opět všechna políčka v ovládacím panelu aplikace, aby mohl uživatel nastavit opět nové hodnoty. A samozřejmě, že tlačítko „Konec“ ukončí celou aplikaci.
4. Ovládání rychlosti simulace, tedy jestli běh simulace bude prováděn v reálném čase nebo mnohonásobně rychleji. To záleží na samotném uživateli aplikace. Jako základ je nastavena jednička, což znamená, že se simuluje v reálném čase. V tomto případě jsou tlačítka zneviditelněná, protože simulace na obrázku 8 zatím není spuštěná. Po stisku tlačítka „Spusť“ se zviditelní pouze políčko „Stop“, protože logicky, když aplikace běží, nebude uživatel znovu klikat na tlačítko „Start“. Po stisku tlačítka „Stop“ se simulace pozastaví a zviditelní se tlačítko „Start“ i políčko pro určení rychlosti průběhu simulace. Stiskem „Start“ se celá simulace znovu rozeběhne z toho místa, kde byla simulace pozastavena. Při stisknutí „Spusť“ by se celá simulace spustila znovu od začátku, proto se při pozastavení simulace pouze zviditelní tlačítko „Start“.

5. Momentálně zneviditelněné tlačítko „Uložit do souboru“ se používá, jak sám název napovídá, pro ukládání do již vytvořeného souboru. Tlačítko se zviditelní, jakmile doběhne celá simulace, protože logicky uživatel bude chtít uložit do souboru všechny nasimulované hodnoty. Soubor se vytvoří ve stejné složce jako samotná aplikace, kam se ukládají jednotlivé simulace pod sebe. Soubor je typu txt pod názvem „Vysledek“.
6. Prozatím prázdné okno, ve kterém se vypisují všechny nasimulované hodnoty výšky hladin jednotlivých nádrží. Ty jsou situovány do sloupců, aby byl výpis přehledný. Samozřejmě nechybí i vypsaný čas, aby bylo uživatelům jasné, jaká je výška hladiny v nádržích v určitém časovém pásmu.
7. Obrázek informující uživatelé o aktuálním zapojení hydraulické soustavy, pro kterou jsou hodnoty nasimulovány. Nechybí také popisky okótovaných částí.
8. Grafické vyobrazení jednotlivých nádrží, které se plní kapalinou. Jedná se o praktickou ukázku plnění nádrží v čase, aby uživatel nebo studenti mohli reálně vidět průběh výšky hladin v jednotlivých nádržích. Zjistí tak jestli některá nádrž nepřetekla nebo do jaké výšky se hladina v nádrži dostane. Vše je graficky znázorněno, kde kapalina v nádrži má příhodně modrou barvu a jakmile nádrž přeteče, tak celé pole zčervená.

Pro jednoduchou ukázku simulace v aplikaci, znázorňuje níže vložený obrázek.



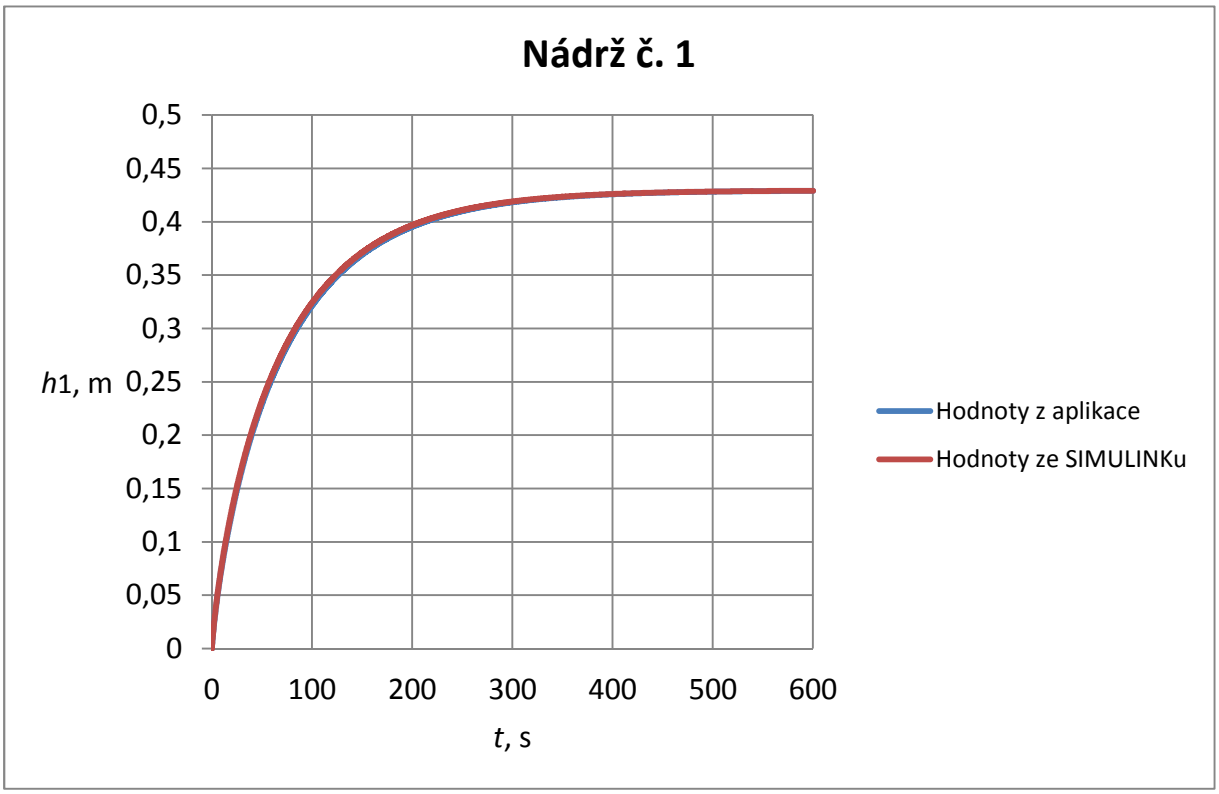
Obrázek 9 – Názorná ukázka simulace v aplikaci

Z obrázku 9 je patrné, jak bylo výše uvedeno, že se textová políčka zneviditelnila, nahoře ve výsledkovém okně se vypsaly hodnoty výšek jednotlivých nádrží v čase a vpravo dole se

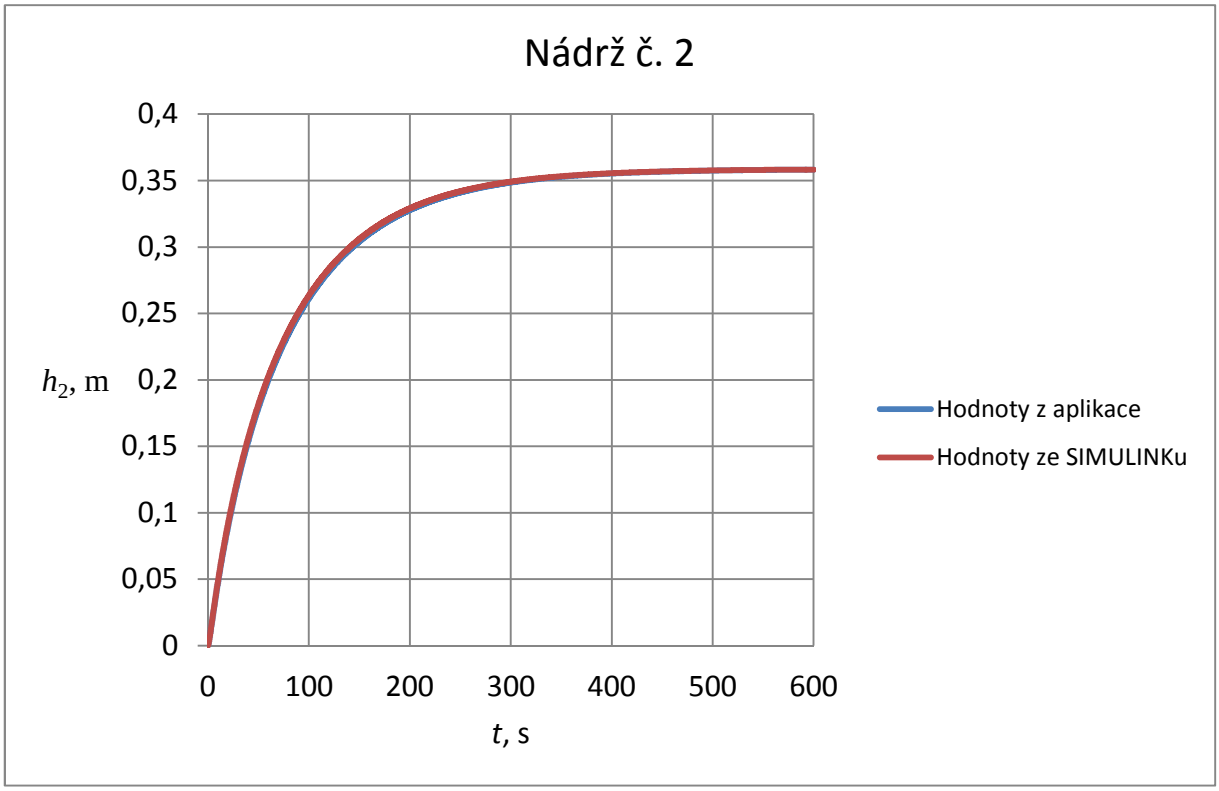
nádrže naplnily kapalinou podle reálných nasimulovaných hodnot. Také se provedlo uložení nasimulovaných hodnot do souboru, o čemž uživatele informuje oznámení vedle tlačítka „Uložit do souboru“. Tím by byla popsána celá aplikace, ze které se hodnoty převezmou k dalšímu zpracování.

2.6 Porovnání nasimulovaných hodnot

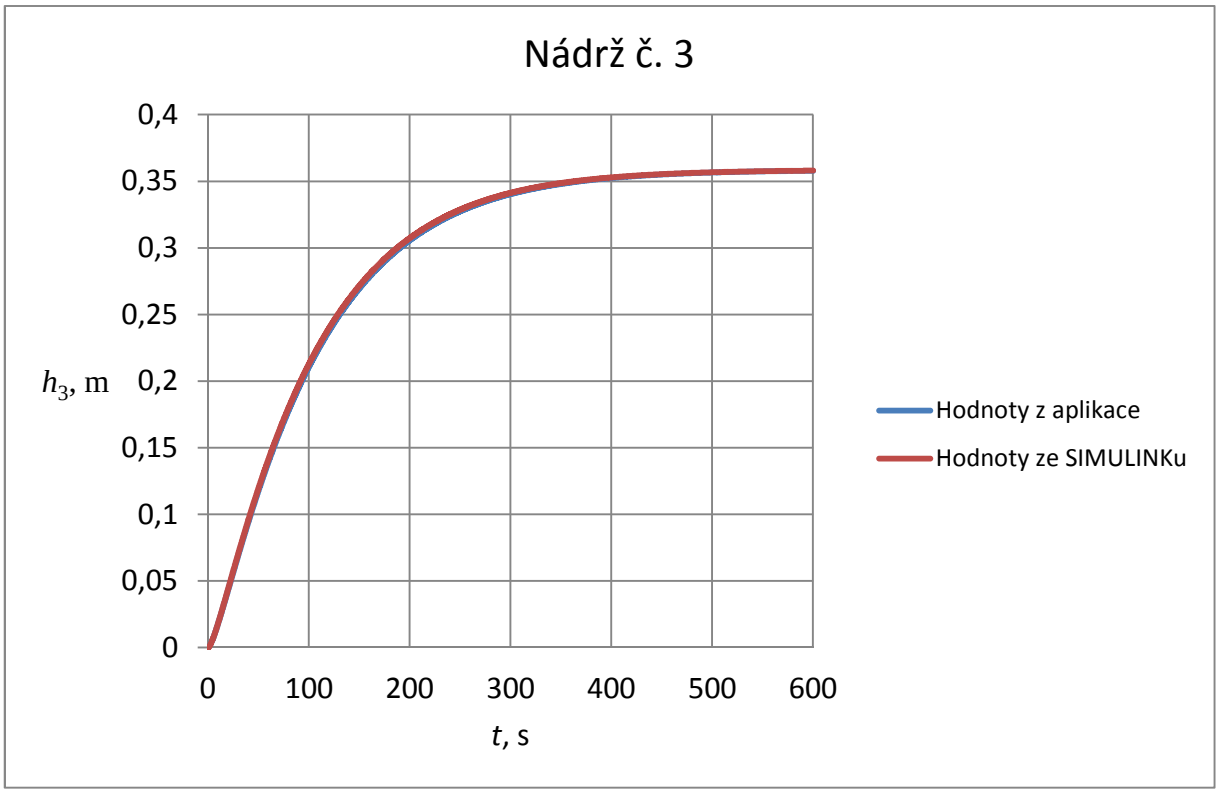
Řešením celé bakalářské práce je porovnání nasimulovaných hodnot z nadstavby SIMULINK a vytvořené aplikace, kde se využívá numerická metoda Runge-Kutta. Pomocí jednotlivých simulací jsou zjištěné hodnoty výšky hladin a za pomocí grafu se zjišťuje, jak velkou odchylkou disponují. Pro tuto práci je zvolen krok času o velikosti 0,2 s. Tento krok byl zvolen kvůli dostatečnému počtu naměřených hodnot, aby grafy měly v podstatě spojitý průběh. Pro celou soustavu je konstantně nastaveno napouštění na hodnotu 2 l/min. Z nasimulovaných hodnot aplikace a nadstavby jsou patrné odchylky, které jsou téměř minimální. Odchylky simulací se pohybují v řádech deseti tisícin metrů. Pro lepší pochopení těchto malých odchylek jsou vloženy grafy, které jsou vyobrazeny na obrázcích 10, 11 a 12.



Obrázek 10 – Graf obou simulací pro nádrž č. 1



Obrázek 11 – Graf obou simulací pro nádrž č. 2



Obrázek 12 – Graf obou simulací pro nádrž č. 3

Díky obrázkům 10, 11 a 12 je zřetelně vidět, že grafy obou simulací jsou totožné a odchylky jsou téměř zanedbatelné. Pro lepší orientaci je dobré si uvést číselné a grafické vyhodnocení odchylek všech nádrží. K tomu by nám mohl posloužit následující vzorec

$$O = \frac{\sum_{i=1}^{M} \left| h_{Si} - h_{Ai} \right|}{M}, \tag{22}$$

kde O – průměrná odchylka hladin nádrží,

M – počet naměřených hodnot,

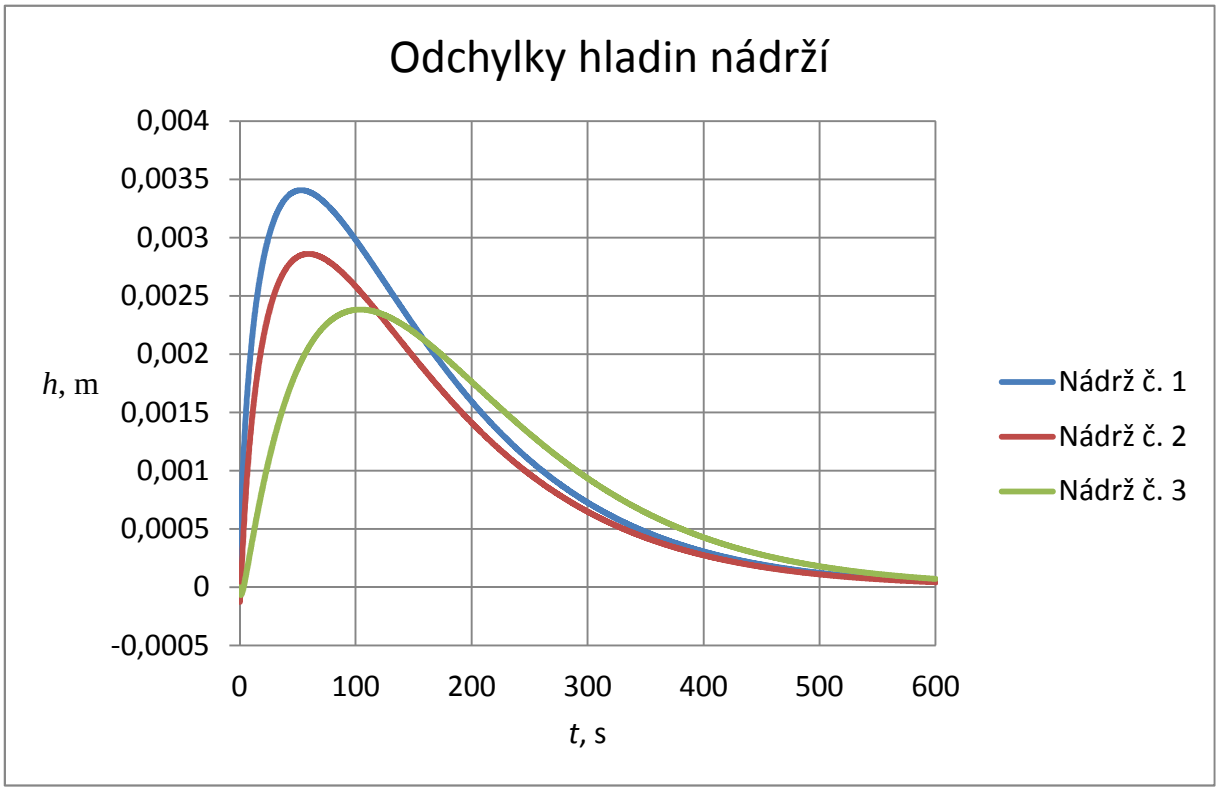
h_{Si} – hladina nádrže v určitém kroku naměřená v SIMULINKu,

h_{Ai} – hladina nádrže v určitém kroku naměřená v aplikaci.

Podle výše uvedeného vzorce jsou spočítány odchylky hladin jednotlivých nádrží, kde průměrné odchylky hladin jsou:

1. h_1 = 0,001186 m,
2. h_2 = 0,001013 m,
3. h_3 = 0,001025 m.

Z výše uvedených hodnot je patrné, že se jedná o odchylky v řádech milimetrů. Pro lepší přehlednost je zobrazen níže na obrázku 13 graf odchylek hladin jednotlivých nádrží.



Obrázek 13 – Graf odchylek hladin jednotlivých nádrží

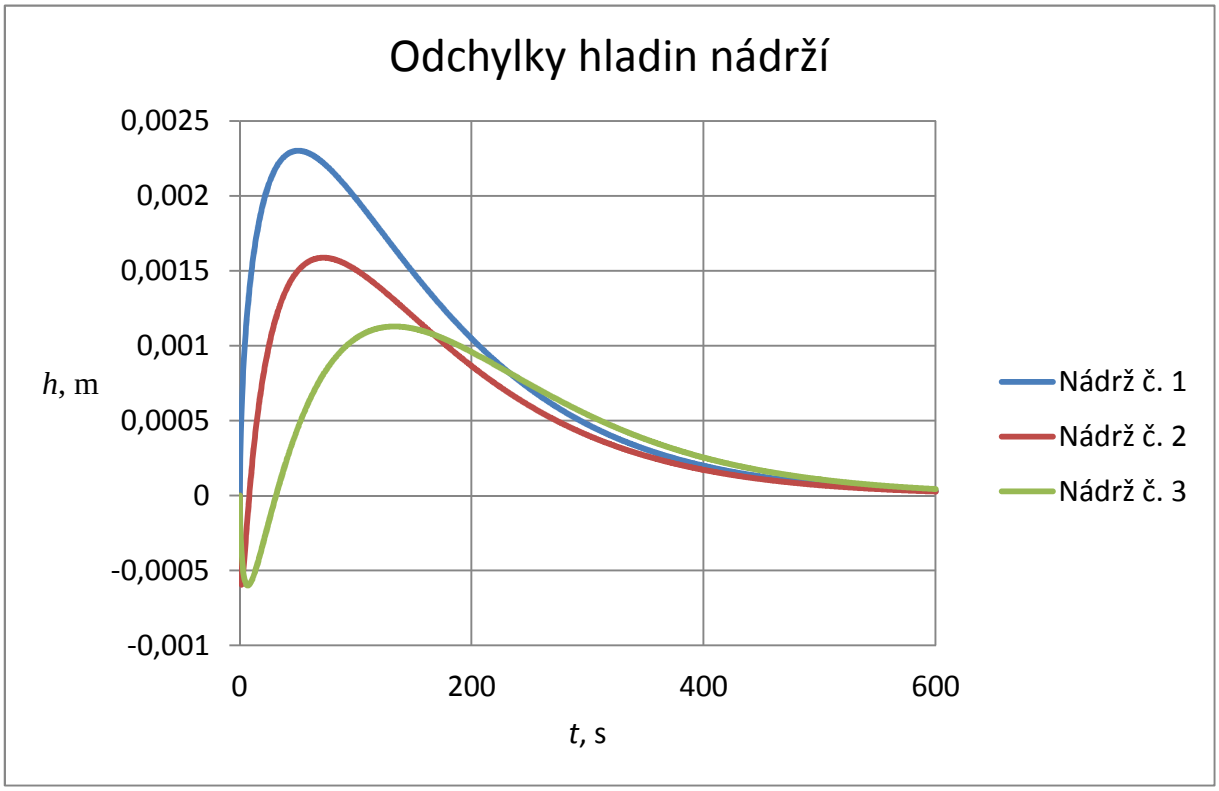
Z grafu na obrázku 13 je zřejmé, že pro nižší hodnoty je odchylka větší a s rostoucím časem a zvyšující se hladinou se odchylka postupně snižuje, čímž se blíží nule. Je to naprosto logické,

protože ze začátku se hladiny nádrží pohybují v řádech milimetrů až centimetrů, kde odchylky hladin nádrží jsou právě v řádech milimetrů. Avšak tyto odchylky dle grafu nejsou nikterak vysoké, kde nejvyšší odchylku zaznamenala nádrž č. 1 a to téměř 3,5 mm.

V rámci práce je také proveden experiment, kde jsou všechny nastavení zachovány až na zvolený krok času, který je zvolen na 0,5 s. Tím se sníží počet naměřených hodnot a při použití vzorce (22) dochází k následujícím průměrným odchylkám hladin jednotlivých nádrží:

1. h_1 = 0,000791 m,
2. h_2 = 0,000576 m,
3. h_3 = 0,000502 m.

Z toho je patrné, že se zvyšováním volitelného kroku času se bude snižovat odchylka. Vysvětlením je, že hraje velkou roli právě daný počet naměřených hodnot, tedy zvolený krok času, protože se zvyšováním počtu hodnot, snižováním kroku času, se také bude zvyšovat odchylka. Podstatě se budou zaznamenávat hodnoty po velmi malých krocích a tím se tak zaznamená více odchylek hladin o vyšší hodnotě. Pro lepší představu je níže na obrázku 14 zobrazen graf odchylek hladin jednotlivých nádrží tohoto experimentu.



Obrázek 14 – Graf odchylek hladin jednotlivých nádrží – experiment

3 Závěr

Cílem bakalářské práce je porovnat chování modelu v naprogramované aplikaci pomocí jazyku JAVA v rozhraní GUI, s modelem dříve vytvořeným v nadstavbě SIMULINK.

Důležitým faktorem je, že musí existovat model, se kterým je možné porovnávat. Proto se nejdříve navržená hydraulická soustava vytvoří v nadstavbě SIMULINK, kde slouží jako vzorový model. Pro získání věrohodných hodnot výšek hladin nádrží se musí na základě diferenciální rovnic soustavy sestavit model. SIMULINK je přímo vytvořený pro tuto práci, a proto vzorový model je sestaven přímo v této nadstavbě. Sestavený model se pak simuluje pořád dokola, dokud výšky hladin nádrží jsou ustálené a mají takové hodnoty, se kterými lze dále pracovat. K zobrazení slouží blok určený pro grafické průběhy.

Dalším krokem je vyhledání vhodné numerické metody pro řešení diferenciálních rovnic popisujících soustavu. Byla vybrána metoda Runge-Kutta. Použitím metody jsou sestaveny konečné rovnice pro aplikaci. Po naprogramování aplikace podle zadání se provádí simulace modelu, která využívá právě již popsanou numerickou metodu. Aplikace má stejné možnosti nastavování parametrů jako v SIMULINKu, ale navíc má graficky znázorněné plnění nádrží, výpis hodnot a také možnost ukládat tyto hodnoty do souborů. Pro porovnání se používají hodnoty obou simulaci, kde se zjistí jejich odchylka. Z hodnot, výpočtu a grafu je patrné, že oba modely mají velice shodné chování, protože nasimulované hodnoty obou simulací jsou téměř totožné. Hodnoty odchylek hladin nádrží se pohybují v řádech milimetrů až desetin milimetrů. Tím se dá říci, že metoda je velmi přesná a lze ji využívat k těmto účelům. Pomocí experimentu je také dokázáno, že se zvyšováním volitelného kroku času se odchylky hladin nádrží snižují.

Po skeptickém začátku se zdá být bakalářská práce velmi zážívná a nabízí mnoho nových zkušeností a také aplikace vytvořená pro simulace hydraulické soustavy je možné využívat jako výukovou aplikaci.

Seznam použité literatury

[1] NEKVINDA, Miloslav, Jiří ŠRUBAŘ a Jaroslav VILD. *Úvod do numerické matematiky*. Praha: SNTL - Nakladatelství technické literatury, 1976, s. 212-217. ISBN L11-B3-IV-41/11731.

[2] RALSTON, Anthony. *Základy numerické matematiky*. Praha: Academia, 1978, s. 219-229. ISBN 104-21-852.

[3] DUŠEK, František. *MATLAB a SIMULINK: úvod do používání*. Vyd. 1. Pardubice: Univerzita Pardubice, 2000, 146 s. ISBN 80-7194-273-1.

[4] HEROUT, Pavel. *Učebnice jazyka Java: úvod do používání*. 5., rozš. vyd. České Budějovice: Kopp, 2010, 386 s. ISBN 978-80-7232-398-2.

[5] Designing a Swing GUI in NetBeans IDE. *NetBeans IDE: The Smarter and Faster Way to Code* [online]. 2013 [cit. 2013-05-11]. Dostupné z: https://netbeans.org/kb/docs/java/quickstart-gui.html#getting_started

[6] *Práce se soubory v Javě*. Brno, 5.4.2011. Dostupné z: http://www.sspbrno.cz/~lenka.hruskova/programovani/java/soubory/Java-prace-se-soubory.pdf

[7] Show Dialog Box in Java - Swing Dialogs. *Rose India Technologies Pvt. Ltd.* [online]. 14.4.2007 [cit. 2013-05-11]. Dostupné z: http://www.roseindia.net/java/example/java/swing/ShowDialogBox.shtml

[8] Programovací jazyk JavaTM: ÚVODNÍ INFORMACE. *Nezávislý studentský informační server Západočeské univerzity v Plzni* [online]. 2013 [cit. 2013-05-11]. Dostupné z: http://v1.dione.zcu.cz/java/uvod.html

[9] Java (24) - úvod do grafiky a GUI. KYSILKA, Pavel. *Linuxsoft.cz* [online]. 2006 [cit. 2013-05-11]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=1184

[10] Java a základy GUI. *Zaachi.com* [online]. 2008 [cit. 2013-05-11]. Dostupné z: http://www.zaachi.com/cs/items/java-a-zaklady-gui.html

[11] FABER, T. Fluid dynamics for physicists. Cambridge: Cambridge University Press, 1995, xxvi, 440 p. ISBN 0-521-42969-2

[12] BEDNAŘÍK, Milan, Miroslava ŠIROKÁ a Jaromír ŠIROKÝ. Fyzika I: Učebnice pro stud. obory středních odb. učilišť. 3. vyd. Praha: SPN, 1991, 212 s. Učebnice pro střední školy. ISBN 80-042-5515-9.

[13] KOTALA, Zdeněk a Petr TOMAN. Kávu, prosím?. *Studentský informační server* [online]. 2001, 4.2.2001 [cit. 2013-05-11]. Dostupné z: http://v1.dione.zcu.cz/java/sbornik/3.html#3_SEC

Příloha A – Ukázka kódu v jazyce JAVA

```
//univerzalni metoda pro vypocet plochy nadrze a prutoku
   private double VypoctiSf(double prumer) {
      double Sf;
      Sf = (Math.PI * Math.pow(prumer, 2)) / 4;
      return Sf;
   }

private double Vypocti(double z, double napust, double pNadrz, double pPrutok1, double pPrutok2, double pPrutok3, double h1, double h2, double h3, int hodnota) {
      double K1, K2, K3, K4;
      double g = 9.81;



switch (hodnota) {
         case 1:

            if ((h1 - h2) == 0) {
               K1 = ((napust - 0) / VypoctiSf(pNadrz));
            } else {
               K1 = Math.abs((napust - (VypoctiSf(pPrutok1) * Math.sqrt(2 * g * Math.abs(h1 - h2)))) / VypoctiSf(pNadrz));
            }
            K2 = Math.abs((napust - (VypoctiSf(pPrutok1) * Math.sqrt(2 * g * Math.abs((h1 + (z * K1 / 2)) - h2)))) / VypoctiSf(pNadrz));
            K3 = Math.abs((napust - (VypoctiSf(pPrutok1) * Math.sqrt(2 * g * Math.abs((h1 + (z * K2 / 2)) - h2)))) / VypoctiSf(pNadrz));
            K4 = Math.abs((napust - (VypoctiSf(pPrutok1) * Math.sqrt(2 * g * Math.abs((h1 + K3) - h2)))) / VypoctiSf(pNadrz));

            h1 += (z / 6) * (K1 + 2 * K2 + 2 * K3 + K4);

            if(h1 >= vNadrz) {
               h1 = vNadrz;
            }

            return h1;
         case 2:

            K1 = Math.abs((VypoctiSf(pPrutok1) * Math.sqrt(2 * g * Math.abs(h1 - h2)) - VypoctiSf(pPrutok2) * (Math.sqrt(2 * g * h2))) / VypoctiSf(pNadrz));
            K2 = Math.abs((VypoctiSf(pPrutok1) * Math.sqrt(2 * g * Math.abs(h1 - (h2 + (z * K1 / 2)))) - VypoctiSf(pPrutok2) * Math.sqrt(2 * g * (h2 + (z * K1 / 2)))) / VypoctiSf(pNadrz));
            K3 = Math.abs((VypoctiSf(pPrutok1) * Math.sqrt(2 * g * Math.abs(h1 - (h2 + (z * K2 / 2)))) - VypoctiSf(pPrutok2) * Math.sqrt(2 * g * (h2 + (z * K2 / 2)))) / VypoctiSf(pNadrz));
```

```
            K4 = Math.abs((VypoctiSf(pPrutok1) * Math.sqrt(2 * g * Math.abs(h1 - (h2 + K3))) - VypoctiSf(pPrutok2) * Math.sqrt(2 * g * (h2 + K3))) / VypoctiSf(pNadrz));

            h2 += (z / 6) * (K1 + 2 * K2 + 2 * K3 + K4);

            if(h2 >= vNadrz) {
               h2 = vNadrz;
            }

            return h2;
         case 3:

            K1 = Math.abs((VypoctiSf(pPrutok2) * Math.sqrt(2 * g * h2) - VypoctiSf(pPrutok3) * Math.sqrt(2 * g * h3)) / VypoctiSf(pNadrz));
            K2 = Math.abs((VypoctiSf(pPrutok2) * Math.sqrt(2 * g * h2) - VypoctiSf(pPrutok3) * Math.sqrt(2 * g * (h3 + (z * K1 / 2)))) / VypoctiSf(pNadrz));
            K3 = Math.abs((VypoctiSf(pPrutok2) * Math.sqrt(2 * g * h2) - VypoctiSf(pPrutok3) * Math.sqrt(2 * g * (h3 + (z * K2 / 2)))) / VypoctiSf(pNadrz));
            K4 = Math.abs((VypoctiSf(pPrutok2) * Math.sqrt(2 * g * h2) - VypoctiSf(pPrutok3) * Math.sqrt(2 * g * (h3 + K3))) / VypoctiSf(pNadrz));

            h3 += (z / 6) * (K1 + 2 * K2 + 2 * K3 + K4);

            if(h3 >= vNadrz) {
               h3 = vNadrz;
            }

            return h3;
         default:
            return 0;
      }
   }
```

Příloha B – CD

Obsah adresáře:

- VladykaL_SimulaceHydraulické_PD_2014.pdf

Složka Simulator – src – Bakalarka:

- Nadrze.java – zdrojový kód
- Voda1.java – zdrojový kód

Složka Simulator – dist:

- Simulator.jar – samotná aplikace