Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Návrh a implementace podpůrného softwarového nástroje pro řešení retenční funkce nádrže

Bc. Tomáš Zachoval

Diplomová práce

2019

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2018/2019**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Tomáš Zachoval**
Osobní číslo:	**I17229**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Návrh a implementace podpůrného softwarového nástroje pro řešení retenční funkce nádrže**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

V teoretické části práce budou popsány možnosti využití informačních technologií v oblasti klimatu se zaměřením na technickobezpečnostní dohled nad vodními díly s ohledem na jejich bezpečnost. Praktická část bude zaměřena na návrh a implementaci podpůrného softwarového nástroje pro řešení retenční funkce nádrže. Softwarový nástroj bude navržen jako multiplatformní aplikace umožňující realizovat vybrané matematické přístup pro numerické řešení bilanční rovnice přítoku, odtoku a akumulace vody v nádrži s podporou vizualizace.

Rozsah grafických prací:

Rozsah pracovní zprávy: **50-60 stran**

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

MEHTA, Rajat, Big Data Analytics with Java: Data analysis, visualization machine learning techniques, California: Packt Publishing, [2017]. ISBN 978-1787288980
SARANG, P. G., Java programming New York: McGraw-Hill Professional, c2012. ISBN 978-0071633604

Vedoucí diplomové práce: **Ing. Jan Fikejz, Ph.D.**
Katedra softwarových technologií

Datum zadání diplomové práce: **22. října 2018**
Termín odevzdání diplomové práce: **18. května 2019**

Univerzita Pardubice Česká republika
L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 17. listopadu 2018

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.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 21. 5. 2019

Tomáš Zachoval

PODĚKOVÁNÍ

Na tomto místě bych rád poděkoval všem přátelům a rodině, kteří při mně v době studia stáli a aktivně mě podporovali.

Jmenovitě děkuji všem členům NinjaAreny za nejlepší prožití školních let. Dále mé poděkování patří panu Ing. Janu Fikejzovi, Ph. D. za vedení v průběhu tvorby této práce a panu Ing. Ladislavovi Roušarovi, Ph. D. za odborné konzultace a testování aplikace.

ANOTACE

Diplomová práce se zabývá návrhem a implementací simulačního nástroje umožňujícího realizovat vybraný matematický přístup pro řešení bilanční rovnice přítoku, odtoku a retence vody v nádrži. Součástí práce je popis matematických metod a technologií které se touto problematikou zabývají.

KLÍČOVÁ SLOVA

Java, JavaFX, MVC, Hráz, Nádrž, Přehrada, Hladina

TITLE

Design and implementation of a supporting software tool to solve of the tank retention function

ANNOTATION

The diploma thesis deals with the design and implementation of a simulation tool allowing to implement a selected mathematical approach for the solution of the balance of the inflow, outflow and water retention in the tank. Part of the thesis is a description of mathematical methods and technologies that deal with this issue.

KEYWORDS

Java, JavaFX, MVC, Dam, Tank, Surface

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

API	Application programming interface
CSS	Kaskádové styly
CSV	Comma separated values
GUI	Graphical User Interface
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
MVC	Model-view-controller
UML	Unified Modeling Language
XML	Extensible Markup Language

ÚVOD

V současné době dochází na celém světě ke klimatickým změnám. S těmito změnami dochází i ke změnám v oblasti přístupu k ochraně proti přívalovým povodním. Pro tyto účely slouží suché nádrže neboli polis. Jedná se o vodní díla jejichž úkolem je chránit lidská sídla, majetek i jejich okolí před ničivým působením povodní. Tyto nádrže rozkládají povodňovou vlnu do delšího časového úseku za pomoci dočasné retence vody. Po skončení povodně jsou nádrže zcela nebo částečně vypuštěny a slouží jako zásobárna užitkové vody.

K maximálnímu využití nádrže a jejího transformačního účinku je potřeba docílit toho, aby se plnila jen v případě povodně. Z těchto důvodů je nutné navrhnout, pokud možno co nejvhodnější parametry pro ovládací a bezpečnostní objekty hráze ve vztahu k očekávané povodňové vlně. Toho lze dosáhnout vhodnou kombinaci matematických přístupů pro výpočty dílčích částí nádrže.

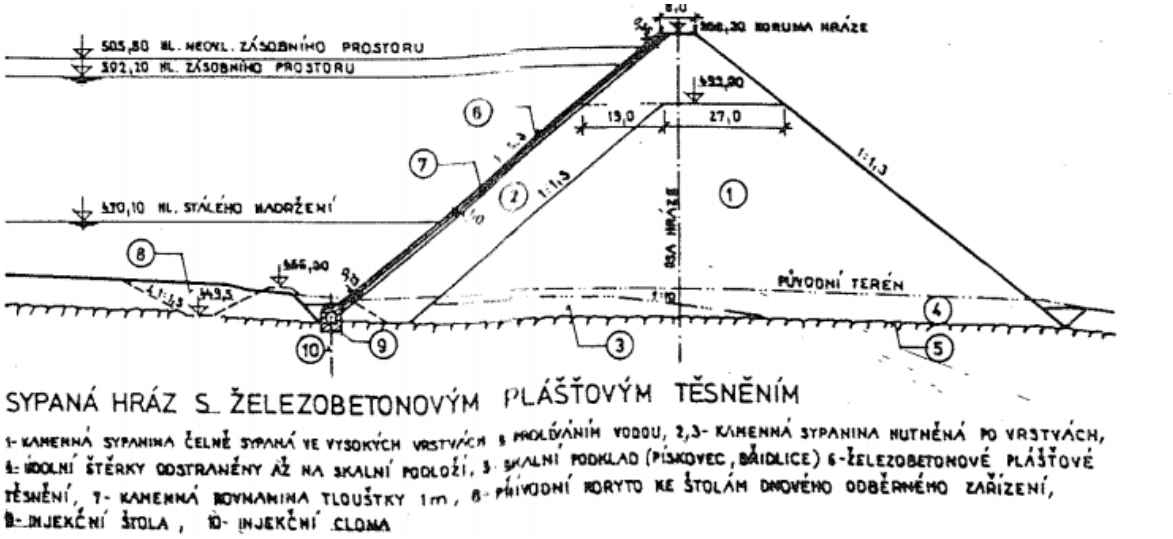
Úkolem této diplomová práce je navrhnout a implementovat právě takový softwarový nástroj, který umožní realizovat vybrané matematické přístupy pro numerické řešení bilanční rovnice přítoku, odtoku a retence vody v nádrži. Pomocí tohoto nástroje lze následně provádět simulaci chování retenční nádrže, a to dle vybraných simulačních scénářů. Tím se zefektivní a zjednoduší práce, která by jinak musela být vykonávána pomocí nástrojů, které nejsou této problematice přímo přizpůsobeny.

V teoretické části jsou popsány všechny matematické výpočty potřebné k docílení pozorovaných hodnot. Dále se zde nachází rešerše vybraných softwarových nástrojů pro řešení této problematiky. Podstatná část práce je věnována návrhu a implementaci výše zmíněné aplikace. V poslední kapitole je popsán experiment, který byl proveden za pomoci nově vytvořeného softwarového nástroje.

1 RETENČNÍ NÁDRŽE

Retenční nádrže jsou objekty, které slouží k zadržení vody. Mohou to být menší nádrže sloužící k zadržení dešťové vody, chránící například kanalizační systémy před zahlcením. Dále to pak mohou být větší přehrady, které zadržují pitnou či užitkovou vodu pro budoucí užití. Retenční nádrže také mohou sloužiti v období povodní pro zadržení povodňové vlny a chrání tím území, které by bylo jinak zatopeno.

Každá přehrada je stavěna na základně, podrobných geologických průzkumů a každé dílo je jedinečně uzpůsobeno prostředí, ve kterém se nachází, přičemž se předpokládá, že tyto objekty budou využívány i více jak sto let. Z těchto důvodů jsou využity nejrůznější techniky a materiály pro výstavbu. Například velmi důležitým aspektem je volba přehradního profilu a typu hráze. Pro upřesnění, typem hráze může být například sypaná hráz s železobetonovým plášťovým těsněním. Tento typ byl vyvinut v USA, kde byl také velmi často využíván. Tato hráz je zobrazena na obrázku č.1. [1]



Obrázek 1: Sypaná hráz s železobetonovým plášťovým těsněním[1]

[1] Zdroj [1]

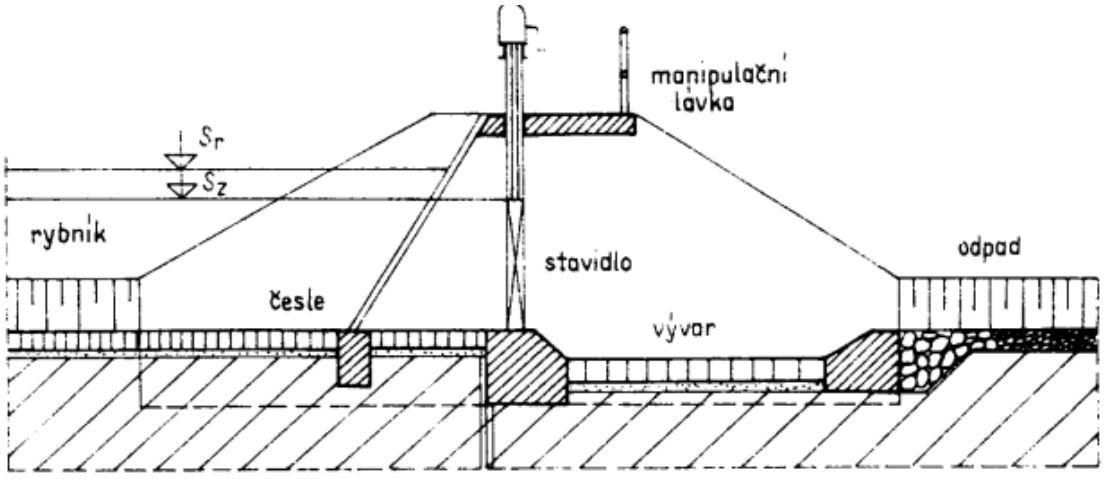
1.1 Funkční objekty hráze

Pro zabezpečení a regulaci vody v nádrži jsou hráze vybavovány funkčními, a bezpečnostními prvky. Práce s těmito objekty, nastavování jejich parametrů a simulace průtoku je jedním z hlavních témat této práce. [1]

1.1.1 Spodní výpusti

Jedním z funkčních objektů hráze jsou spodní výpusti. Jedná se o ovládací objekt, kterým musí být každá hráz bezpodmínečně vybavena. Jejich úkolem je převádět běžné průtoky, prázdnit nebo plnit nádrž. Slouží také k úplnému vypuštění nádrže, a to včetně odvodnění dna.

Součástí je vtokový objekt osazený česlemi. Spodní výpustí zpravidla bývá ocelové potrubí. Toto potrubí je umístěno dle návrhu projektu. Například v již zmiňovaných sypaných přehradách je často umístěno v samostatné štole. Spodní výpust obsahuje minimálně tři uzávěry, a to jeden hlavní a dva provizorní. Hlavní uzávěry se dělí na jednotlivé typy. Jsou to klapkový, rozstřikovací, segmentový, stavidlový. Schéma výpusti je možné si prohlédnout na obrázku 2. [2]



Obrázek 2: Schéma výpusti[2]

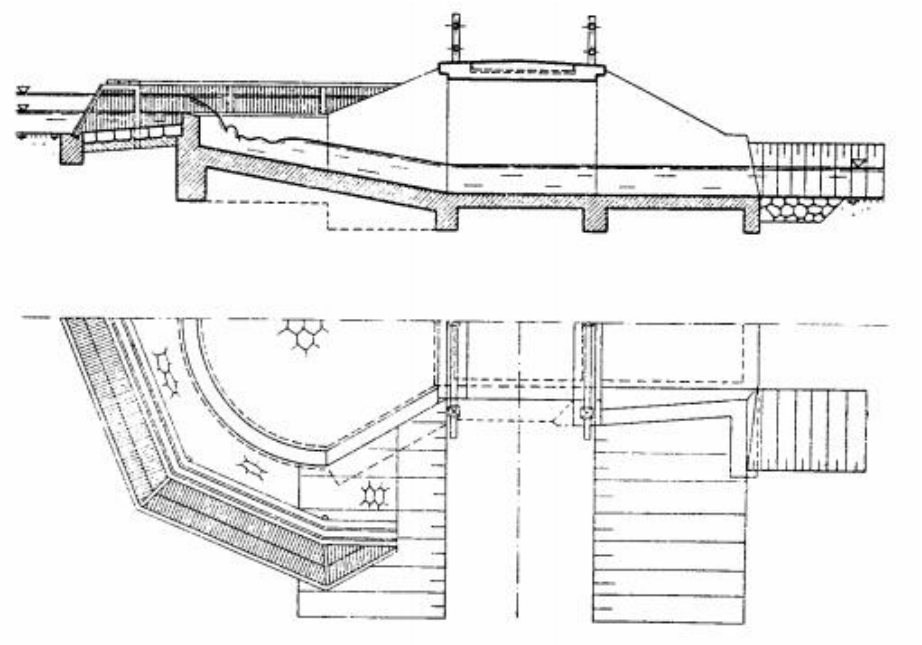
[2] Zdroj [4]

1.1.2 **Bezpečnostní přelivy**

Druhým typem funkčních objektů jsou objekty bezpečností. Mají za úkol převádět povodňové průtoky. Tím chrání proti přelití hráze, které by mohlo způsobit poškození přehrady a jejího okolí.

Přelivy se dělí dle přelivné hrany na hrazené nebo nehrazené. Hrazené přelivy jsou rozděleny na další podtypy. Jsou to přímé, boční, šachtové, sdružené objekty, nouzové, kašnové. Příklad posledního jmenovaného typu přelivu je zobrazen na obrázku 3.

Před přelivem také bývají často vystavěny česle, které zamezují jeho ucpání. [3]



Obrázek 3: Kašnový přeliv[3]

1.2 **Kapacita nádrže**

Každá nádrž má omezenou kapacitu. Na objemu vody v nádrži mají vliv různé aspekty zejména pak počasí. Před výstavbou je tedy vhodné tyto aspekty pozorovat a zjistit jaký je maximální přítok vody i za extrémních podmínek a jakým způsobem bude probíhat plnění přehrady a její vodní průtok. Na základě těchto pozorování jsou voleny bezpečností prvky přehrady. Další kapitoly se zabývají matematickými operacemi a vzorci, potřebnými pro docílení výše zmíněných pozorování.

[3] Zdroj [3]

2 REŠERŠE NÁSTROJŮ PROBLEMATIKY PRŮTOKU

Tato kapitola pojednává o nástrojích, které jsou v současné době používány pro výpočet problematiky průtoku retenčních nádrží. Jsou zde zmíněny jejich výhody, a především nevýhody vedoucí k důvodu pro vytvoření nového softwaru.

Požadavky na nový software i aktuálně používané nástroje, byly konzultovány s externím specialistou v oboru. Tímto konzultantem byl současný jednatel firmy VHRoušar, Ing. Ladislav Roušar, Ph.D. Jeho firma se specializuje na vodohospodářské stavby, a to jak v projekční činnosti, tak i v oblasti stavebního dozoru či zhotovení hydrotechnických výpočtů a studií. Firma má za sebou spoustu úspěšných projektů, jak je možné se dočíst na jejích webových stránkách. Doktor Roušar tak sehrál významnou úlohu při konzultacích v jak období vývoje, tak následně při verifikaci a validaci finální podoby softwarového nástroje.[18]

2.1 Matlab

Matlab je multiplatformní skriptovací programovací jazyk, který je součástí vlastního programového prostředí. Byl vyvinut profesorem Clevem Molerem na univerzitě v Novém Mexiku, odkud se následně šířil i na další univerzy. V roce 1985 byl přepsán do jazyka C a byla vydána jeho oficiální verze firmou Mathworks, která na něm do dnešní doby pracuje. Matlab je prostředí jenž umožňuje řešit různé matematické, fyzikální a další problémy. V jeho prostředí je možné vytvořit komplexní programy pro řešení úloh s různými vstupními parametry. Umožňuje zobrazení grafů i dalších vizuálních komponent. Cekově je možné konstatovat, že tento software je jedním z nejlepších matematických softwarů současnosti. [16]

Práce v Matlabu není pro člověka vzdělaného v informačních technologiích příliš složitá, nicméně je minimálně nutné naučit se syntaxi tohoto jazyka a mít alespoň pokročilé znalosti v algoritmizaci. Výuka tohoto jazyka bývá součástí vysokoškolského vzdělání v technických oborech, nicméně není pravidlem, že by výstupní zkušenosti byly vždy dostatečné pro zpracování problematiky retenčních nádrží. Jednou z hlavních nevýhod Matlabu je jeho vysoká cena.

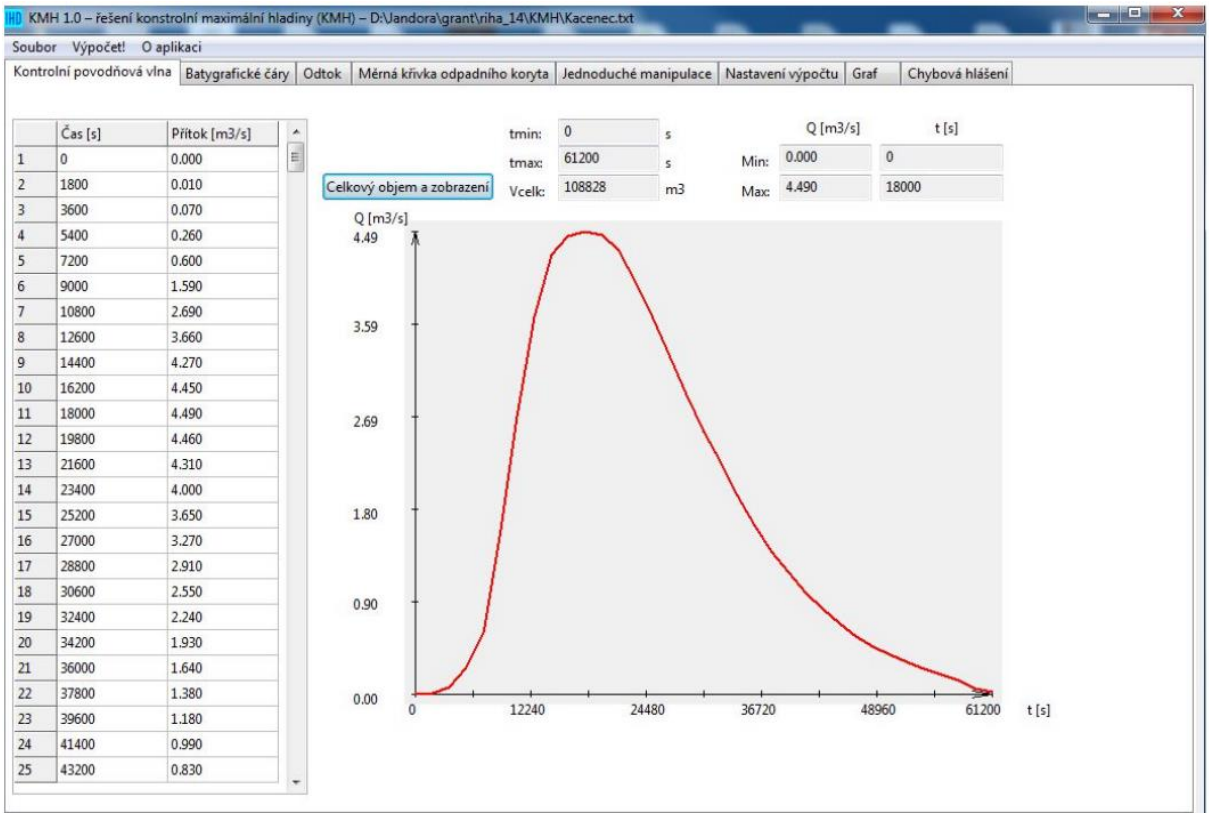
2.2 Tabulkové procesory

Dalšími nástroji, které je možné využít pro danou problematiku jsou nejrůznější tabulkové procesory. Mezi tyto nástroje patří například Microsoft Excel, OpenOffice a podobné editory od konkurenčních firem. Tyto nástroje umožňují výpočty i vizualizaci dat, které jsou umístěny v buňkách. Buňky mohou obsahovat konstanty, vzorce a funkce podporované editory. Práce

s těmito editory není příliš složitá pro běžného uživatele. Nicméně využití v komplexnějších projektech není zcela vhodné, a to zejména pro její časovou efektivitu. Uživatel do jednotlivých buněk přistupuje ve většinou pomocí myši, a pokud jich je velké množství lze s nadsázkou říci, že se takzvaně ukliká. Výpočetní funkce jsou přístupné všem uživatelům a je zde vysoká šance na vznik chyby.

2.3 KMH software

Jedná so o software řešící kontrolu maximální hladiny, který vznikl v roce 2015 na Vysokém učení technickém v Brně. Software umožňuje výpočet průtoků na základě přelivu i výpusti, nicméně má spoustu nedostatků. Za zmínku stojí například, neumožnění zadání více druhů přelivů či obsažení malého množství vztahů pro výpočet součinitele přepadu. Náhled této aplikace je na obrázku 10. [17]



Obrázek 4: KHM software povodňová vlna[4]

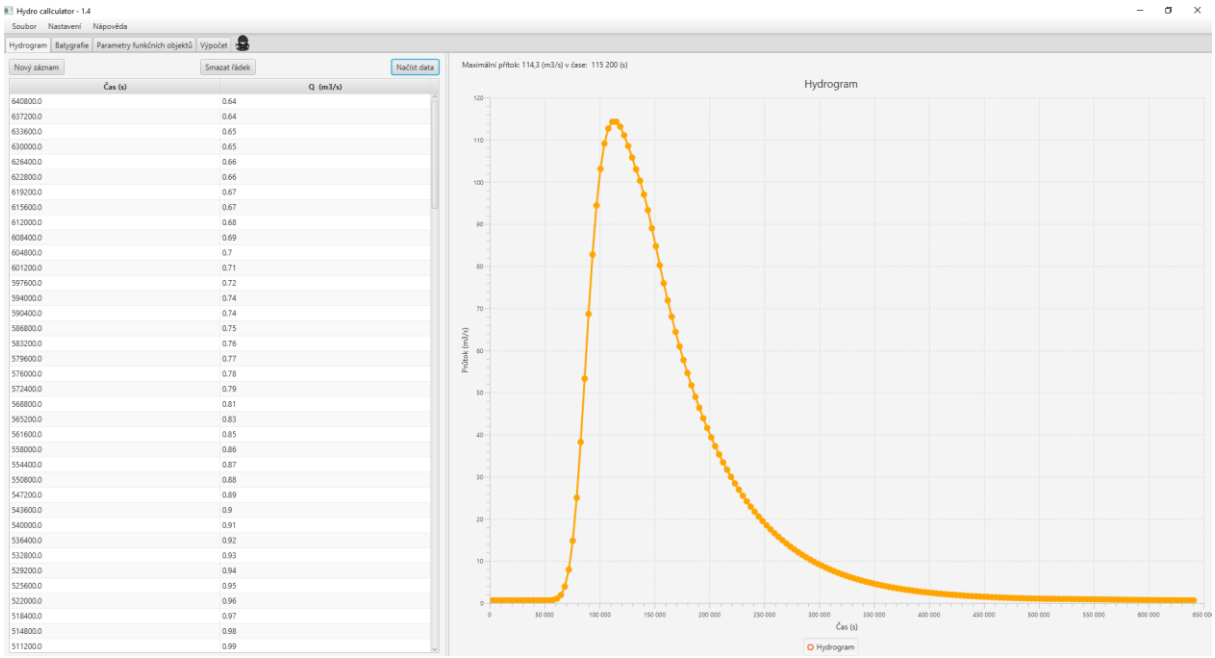
[4] Zdroj [19]

3 POSTUPY A VÝPOČTY

Tato kapitola se zabývá postupem a výpočty, kterými lze docílit simulace stavu různých veličin které jsou pozorovány na vodním díle v konkrétním čase. Mezi tyto hodnoty patří objem vody nacházející se v nádrži, přítok, odtok z jednotlivých funkčních a bezpečnostních objektů, suma těchto odtoků a výška vodní hladiny. Postup je sepsán ve stejném pořadí, ve kterém postupuje uživatel zadávající hodnoty do softwaru, jenž je součástí této diplomové práce.

3.1 Hydrogram přítoku

Hydrogram přítoku tvoří množina dat dvou na sobě závislých parametrů. Je to čas v sekundách a objemu vody v m^3/s. Po vizualizaci tato data tvoří graf, který symbolizuje povodňovou vlnu, jak je vidět na obrázku č. 4.



Obrázek 5: Hydrogram přítoku

V aplikaci data z hydrogramu simulují přítok vody v pozorovaném čase. Software umožňuje zobrazit bod ukazující v jakém čase dosahuje povodňová vlna nejvyššího objemu přítoku. Data umožňuje aplikace importovat ze souboru s standardizovaným formátem CSV. Zdroje pro hydrogram je možné získat z hydrotechnických výpočtů. Tímto se ale tato práce nezabývá, je možné najít například v diplomové páci pana Romana Pitky *Návrh víceúčelové nádrže v k.ú. Popovice* kde se v páté kapitole těmito výpočty zabývá. [6]

3.2 **Batygrafické křivky**

Tyto křivky patří mezi hlavní podklady popisující tvar údolí, ve kterém se hráz nachází. Pro potřeby výpočtů jsou tato data stěžejní. Obsahují následující veličiny:

H (Z) nadmořská výška hladiny v systému Balt (m n. m.)

V objem (m^3)

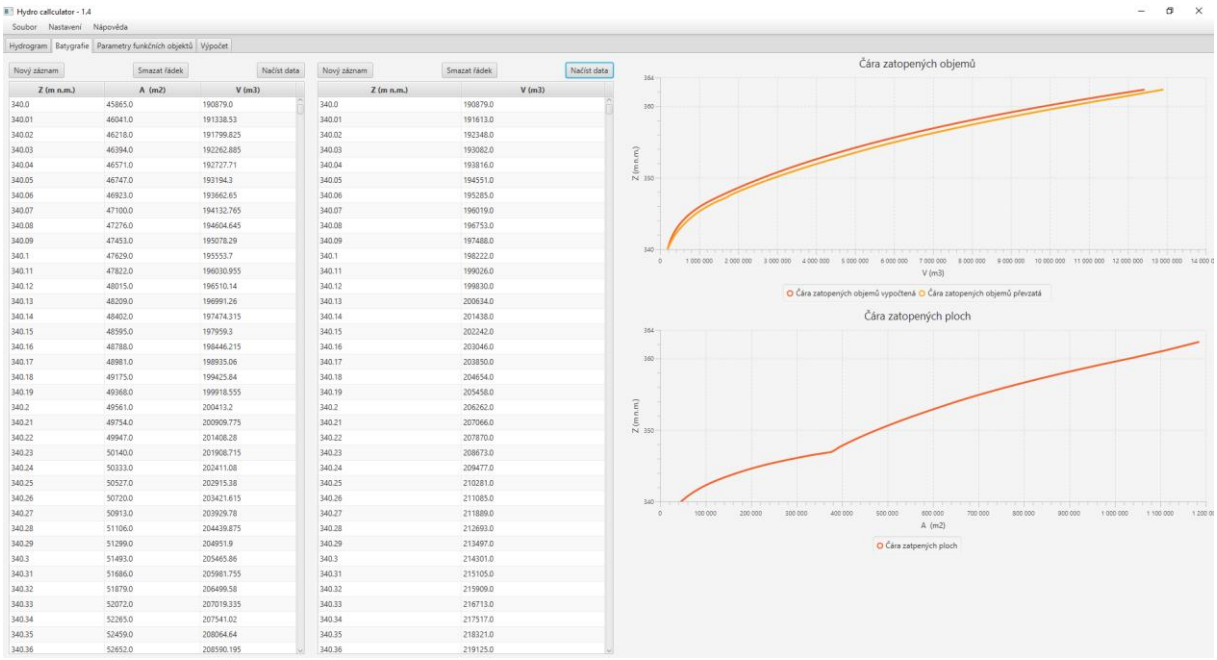
A plocha (m^2)

Výšku hladiny, a plochu je nutné vždy importovat. Aplikace opět umožňuje import dat ze souboru ve formátu CSV. Z těchto dvou hodnot je za pomocí lichoběžníkového pravidla

$$dV(i+1) = H(i+1) - H(i)\frac{A(i) + A(i+1)}{2}$$

$$V(i+1) = dV(i+1) + V(i)$$

a výchozí hodnoty objemu, kterou je nutné zadat jako parametr, vypočten objem pro jednotlivé výšky hladiny. Množina těchto hodnot tvoří tzv. čáru zatopených objemů vypočtenou. Pokud je objem importován ze souboru, nebo ručně zadán, tak množina objemu a výšky hladiny tvoří čáru zatopených objemů převzatou. Ve výsledné kalkulaci je možné vybrat, která čára se zahrne do výpočtů. Hodnoty plochy a výšky hladiny tvoří čáru zatopených ploch. Tato čára „*vyjadřuje závislost zatopené plochy na hloubce zadržené vody, jako podklad slouží podrobné zaměření budoucí zátopy nádrže“*. [6] Všechny výše zmíněné, čáry jsou v aplikaci zobrazeny v tabulce a grafech. Snímek obrazovky z batygrafie se nachází na obrázku č. 5. [5]



Obrázek 6: Čáry zatopených objemů a čára zatopené plochy

3.3 Funkční objekty

Funkční objekty v přehradě jsou objekty, které jsou účelně umístěny přímo v hrázi nebo její bezprostřední blízkosti. Aplikace, která je součástí této diplomové práce pracuje se dvěma typy takovýchto objektů. Jsou to až tři spodní výpusti až tři bezpečnosti přepady.

3.3.1 Spodní výpusť

Jedná se o hlavní ovládací objekt přehrady. Více informací z technického hlediska o tomto objektu zpracovává předchozí kapitola.

Hlavní funkcí je zjistit objem vody který za určité hladiny a v určitém čase protýká spodní výpustí z nádrže. Pro tento výpočet jsou potřebné následující veličiny:

D	průměr výpusti (m)
Z	kóta dna výpusti na výtoku (m n. m.)
L	délka potrubí spodní výpusti
α	coriolisovo číslo (bezrozměrné)

λ	součinitel tření (bezrozměrné)
ε_i	vliv konstrukce (bezrozměrné)
Sv_i	plocha zůžení (m^2)
Sv	průměr potrubí výpusti
Zh	přetlakový výška (m)
H	aktuální hladina (m n. m.)
μ	součinitel výtoku (bezrozměrné)
Q	průtok výpusti (m^3)
g	gravitační zrychlení (9.81)
n	součinitel drsnosti potrubí (bezrozměrné)

Postup výpočtu je následující. Uživatel zadá vstupy, jež aplikace bezpodmínečně vyžaduje. Je to většina zmíněných parametrů až na přetlakovou výšku aktuální hladinu a průtok. Přetlaková výška je vypočítána z následujícího vztahu.

$$Zh = H - Z + \frac{D}{2}$$

Vstupem pro aktuální výšku hladiny jsou data z batygrafických křivek. Součinitel přepadu je možné zadat ručně, nebo dopočítat. Vztah je následující.

$$\mu = \frac{1}{\sqrt{\alpha + \lambda \frac{1}{D} \sum_{i=1}^{n} n_i \varepsilon_i \frac{Sv}{Sv_i}}}$$

Pro tento vztah je ještě nutné znát parametr *Sv*, který je možné vypočítat z průměru výpusti následovně.

$$\mathrm{Sv} = \frac{\pi D^2}{4}$$

Parametr λ je opět možné zadat ručně nebo dopočítat ze vztahů

$$R = \frac{\frac{\pi Sv^2}{4}}{2\pi\frac{Sv}{2}}$$

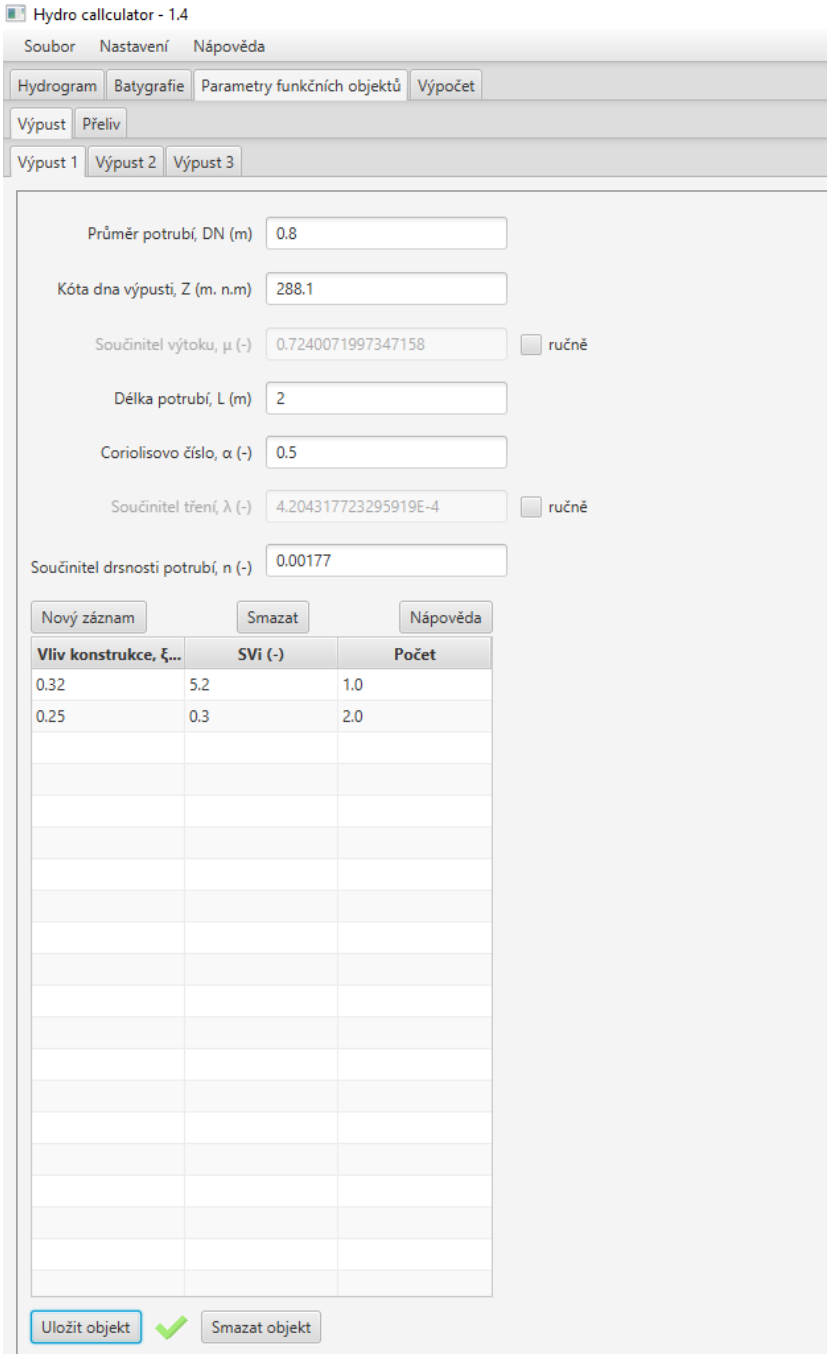
$$\mathrm{C} = \frac{1}{nR^{\frac{1}{6}}}$$

$$\lambda = \frac{8g}{c^2}$$

Po získání předchozích hodnot lze dopočítat výsledný průtok spodní výpusti pro aktuální hladinu ze vztahu

$$Q = \mu\, Sv\sqrt{2g\ Zh}$$

V aplikaci je možné zadat až tři takovéto výpusti, jak bylo zmíněno výše. Ukázka, jakým způsobem ji v aplikaci lze zadávat je vidět na obrázku 6. [5][6]



Obrázek 7: Snímek obrazovky sekce zadání výpusti

3.3.2 **Bezpečnostní přeliv**

Bezpečnostní přeliv je hlavním bezpečnostním objektem bránícím v přetečení hráze. Popis tohoto objektu se opět nachází v předešlé kapitole.

Úkolem aplikace v sekci *přelivy* je stanovit parametry pro vypočet průtoku vody v přelivu. Výpočet je v tomto případě možný více způsoby a je závislý na více parametrech. Jmenovitě je to například výška vodní hladiny nebo typ zaoblení koruny přepadu. V následujícím odstavci jsou vypsány veličiny potřebné k výpočtu.

B	šířka přelivu (m)
b_0	účinná šířka přelivu (m)
Zp	kóta přelivu (m n. m.)
ξ	vliv tvaru pilířů (bezrozměrné)
n	počet ξ (bezrozměrné)
m	součinitel přepadu (bezrozměrné)
H	aktuální hladina (m n. m)
h	přepadová výška (m)
hn	návrhová přepadová výška (m)
r	poloměr zaoblení přelivu hrany (m)
t	tloušťka přelivu (m)
P	výška přelivu mezi dnem nádrže (m)
Ks	Nikuradseho ekvivalentní písková drsností výška (bezrozměrné)
i	sklon koruny přelivu (m)
so	Výška recirkulační oblasti na koruně přelivu bez zaoblení (m)
Ct	vliv relativní tloušťky přelivu (bezrozměrné)
Cks	vliv drsnosti koruny přelivu (bezrozměrné)
Cp	vliv relativní výšky přelivu (bezrozměrné)
Ci	vliv sklonu koruny přelivu (bezrozměrné)
Crus	vliv zaoblení návodní hrany koruny přelivu (bezrozměrné)

Q	průtok přepadem (m^3)
Cd	výsledný součinitel průtoku (bezrozměrné)
M	výsledný součinitel přepadu (bezrozměrné)
a	parametr mocninné funkce nebo polynomu(bezrozměrné)
b	parametr mocninné funkce nebo polynomu(bezrozměrné)
c	parametr polynomu (bezrozměrné)
d	parametr polynomu (bezrozměrné)
e	parametr polynomu (bezrozměrné)

Další odstavce jsou věnovány postupům výpočtů průtoku přelivu z pohledu uživatele aplikace. Ten nejdříve zadá základní vstupní parametry ξ, *n*, *B* a *Zp*. Následně vybere vztah, jakým bude výsledný průtok počítat a zvolí výšku hladiny, od které se bude daným způsobem počítat. Pokud je vyplněna pouze první alternativa výpočet se provádí pro všechny výšky hladiny stejně.

Výpočet začíná pro všechny typy výpočtů stejně. Nejdříve je potřeba vypočítat přepadovou výšku *h*. To je možné z následujícího vztahu.

$$h = H - Zp$$

Dále je zapotřebí vypočítat účinnou šířku přepadu b_0 ze vztahu.

$$b_0 = B - 0{,}1\ \varepsilon\ n\ h$$

Výsledek průtoku přepadu *Q* je možné dosáhnout více způsoby. Prvním způsob je pomocí mocninné funkce s parametry *a*, *b* vyjádřené následujícím vztahem.

$$Q = a\ h^b$$

Druhým způsobem je výpočet na základně polynomu n-tého stupně kdy *a*, *b*, *c*, *d* a *e* jsou vstupní parametry.

$$Q = a + b\ h + c\ h^2 + d\ h^3 + e\ h^4$$

Poslední možností je pomocí součinitele přepadu, jehož vztah je následující.

$$Q = m\, b_0 \sqrt{2g}\, h^{\frac{3}{2}}$$

Pro použití poslední možnosti je nutné vypočítat právě součinitel přepadu m. Zde je opět více možností, jak této hodnoty dosáhnout. Prvním způsobem je zadání této hodnoty ručně z již předem připravených hodnot. Druhým způsobem je použití rovnice, kterou navrhl Rehbock v roce 1929. Vztah této rovnice je následující. [7]

$$m = \frac{2}{3}\left(0{,}312 + \sqrt{0{,}3 - 0{,}01\left(5 - \frac{h}{r}\right)^2 + 0{,}09\,\frac{h}{P}}\right)$$

Dalším způsobem je výpočet na základě rovnice navržené pány Rouve a Indlekofer v roce 1974. [8]

$$m = \frac{2}{3}\left(0{,}9444 + 0{,}35497\frac{h}{r} - 0{,}10791\left(\frac{h}{r}\right)^2 + 0{,}010309\left(\frac{h}{r}\right)^3\right)^{\frac{3}{2}}$$

Jedním ze způsobů je také použít rovnici dle Smetany (1967), která počítá s proudnicovou přelivovou plochou.

$$m = 0{,}499 * \left(0{,}63 + 0{,}37\sqrt{\frac{h}{hn}}\right)$$

Mezi možnosti, jak vypočítat součinitel přepadu patří také vztah dle Bazina (1898) počítající s přepadem přes ostrou hranu.

$$m = \left(0{,}405 + \frac{0{,}003}{h}\right) * \left(1 + 0{,}55\left(\frac{h}{h+P}\right)^2\right)$$

Dále pak vztah navržený Starou v roce 2003. [9]

$$m = 0{,}349494 + 0{,}307084 \frac{\frac{h}{P}}{\left(\left(\frac{h}{P}\right)^{1{,}62877} + \left(\frac{P}{2r}\right)^{-1{,}45694} + 0{,}223402\right)}$$

Poslední možností výpočtu součinitele přepadu na základě vztahů navržených Zachovalem z roku 2014 zaměřených na přepady s šikmou korunou.[10]

Zde jsou na výběr další tři alternativy výpočtu průtoku *Cd*. Všechny tyto alternativy obsahují parametr vliv relativní tloušťky přelivu C*t* s následujícím vztahem.

$$Ct = \frac{\left(1 + \frac{h}{t}\right)^3}{\left(1 + 0{,}75\frac{h}{t}\right)^3}$$

První alternativa zahrnuje vliv relativní tloušťky přelivu *Ct*, vliv relativní výšky přelivu *Cp* se vztahem

$$Cp = 1{,}03 + 0{,}045 \ln\left(\frac{h}{P}\right)$$

a vliv drsnosti koruny přelivu *Cks* s následujícím vztahem.

$$Cks = \left(1 - 0{,}017\frac{t}{h}\sqrt[3]{\frac{ks}{h}}\right)^{1{,}5}$$

Výsledný vztah pro výpočet průtoku *Cd* pro první alternativu má vztah následující.

$$Cd = 0{,}845\ Ct\ Cp\ Cks$$

Druhá alternativa zahrnuje veličiny vliv relativní tloušťky přelivu *Ct* a vliv sklonu koruny přelivu *Ci* jejíž vztah je

$$Ci = 1{,}387 * (0{,}52 - 1{,}19i)^{0{,}5}$$

Výsledný vztah pro výpočet průtoku *Cd* pro druhou alternativu odpovídá vztahu

$$Cd = 0{,}845\ Ct\ Ci$$

Poslední třetí alternativa zahrnuje opět vliv relativní tloušťky přelivu *Ct*, dále pak vliv zaoblení návodní hrany koruny přelivu *Crus* se vztahem

$$Crus = 1{,}126\left(1 - 3\left(\frac{so}{h} - \frac{r}{h}\right)\right)^{0{,}5}\left(1 + \frac{3}{2}\left(\frac{so}{h} - \frac{r}{h}\right)\right)$$

a vliv drsnosti koruny přelivu *Cks*. Výsledný vztah pro výpočet součinitele průtoku třetí alternativy je tedy následující.

$$Cd = 0{,}845\ Ct\ Crus\ Cks$$

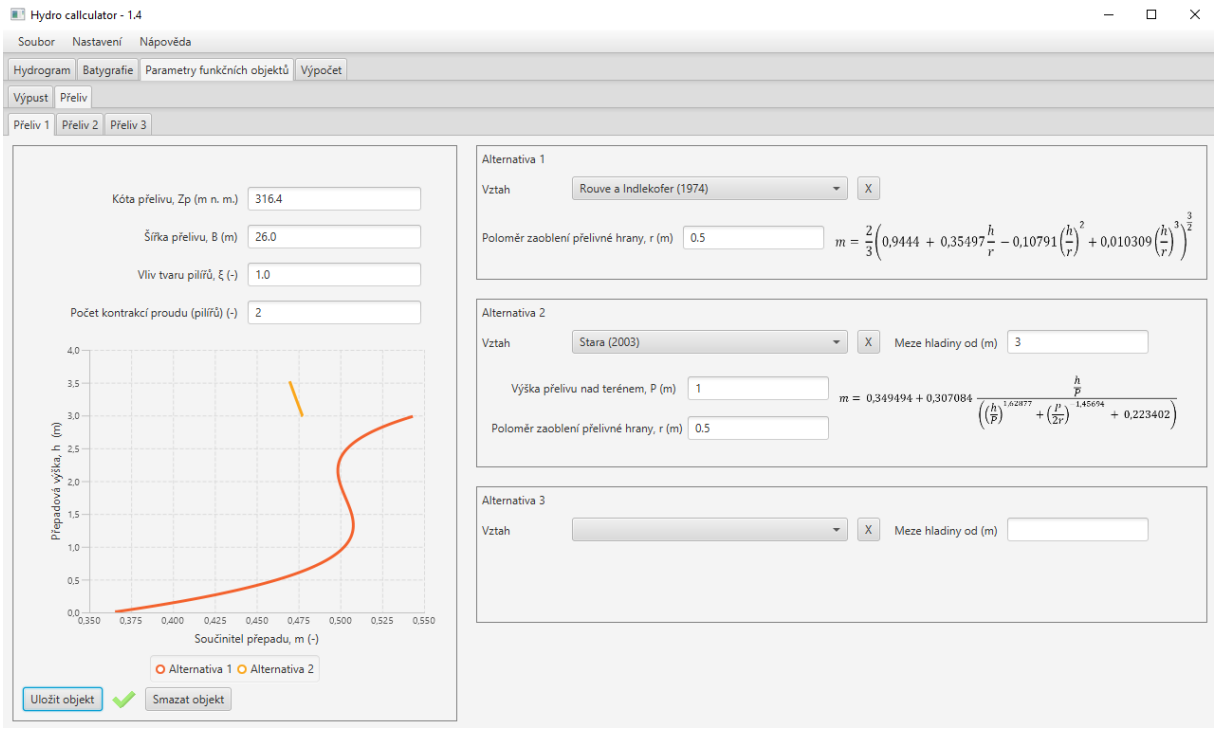
Po získání hodnoty *Cd* je potřeba je potřeba vypočítat součinitel přepadu *M* se vztahem

$$M = Cd\left(\frac{2}{3}\right)^{1{,}5}(9{,}81)^{0{,}5}$$

z něhož se vypočítá výsledný součinitel přepadu *m*, který je dosazen do rovnice pro výpočet průtoku. Vztah pro výpočet *m* je následující.

$$m = \frac{M}{(9{,}81)^{0{,}5}}$$

Aplikace umožňuje zadat až tři takovéto přelivy s různými parametry. Náhled snímku obrazovky se záložkou, ve které je možné zadat vstupní parametry přepadu je na obrázku číslo 7.



Obrázek 8: Snímek záložky přelivu

3.4 Výstupní data

Výstupem z aplikace jsou data obsahující hodnoty veličin pozorovaných u nádrže v konkrétním čase. Tyto velečiny jsou následující.

t_0	časový krok (s)
t	čas (s)
Q	přítok (m^3/s)
H	výška hladiny (m n. m)
V	objem (m^3)
O	odtok jednotlivých funkčních objektů (m^3/s)
dV/dt	diference objemu přítoku a odtoku (m^3/s)
dV	diference objemu přítoku a odtoku v pozorovaném časovém kroku (m^3)

3.4.1 **Postup výpočtu**

- Krok 1: Časový krok (t_0) je stanoven z parametru, zadaného uživatelem aplikace před zahájením výpočtu. V praxi je tento krok 1–100 sekund. Jednotlivé časy jsou počítány tímto vztahem

$$t_{i+1} = t_i + t_0$$

- Krok 2: Přítok (Q) je převzat z hydrogramu na základě pozorovaného času. Pokud se daný čas v hydrogramu nenachází pak je interpolován z nejbližších hodnot.

- Krok 3: Objem (V) v čase 0 je objem stanoven z batygrafických křivek, na základě výšky hladiny. V dalších časech je vypočten z diference průtoku a odtoku.

- Krok 4: Výška Hladiny (H) je b čase 0 stanovena z parametru, zadaným uživatelem aplikace před zahájením výpočtu. V dalších časech je stanovena z batygrafických křivek, na základě aktuálního objemu. Pokud se daná hodnota objemu v batygrafických křivkách nenachází pak je interpolován z nejbližších hodnot.

- Krok 5: Odtok výpustěmi (O_v) je vypočítán pro na základě parametrů konkrétní výpusti a výšky hladiny vody v aktuálním čase.

- Krok 6: Odtok přelivy (O_p) je vypočítán pro na základě parametrů konkrétního přelivu a výšky hladiny vody v aktuálním čase.

- Krok 7: Výpočet diference průtoku v časovém kroku (dV) je stanovena poměrem sumy odtoků z funkčních objektů ($\sum O$) a přítoku (Q). Tato diference je následně vynásobena hodnotou časového kroku (t). Záporná diference značí že výtok je větší než přítok a naopak. Vztah pro výpočet diference průtoku je následující

$$\sum O = \sum O_v + \sum O_P$$

$$dV = \left(\sum O - Q\right)\Delta t$$

- Krok 8: Objem pro další krok (V_{i+1}) je součet aktuální hodnoty objemu (V) a diference průtoku v pozorovaném časovém kroku (dV).

$$V(i+1) = V(i) + dV$$

- Krok 9: Ukončení výpočtu, proběhne úspěšně pokud byly zahrnuty všechny hodnoty časů v hydrogramu. Může být také ukončen chybou, a to v případě že hodnota obejmu vody je větší než maximální objem uvedený v batygrafických křivkách. V těchto případech není možné dané hodnoty pomocí interpolace dopočítat a je zapotřebí aby byly přidány do batygrafie.

3.4.2 **Specifikace výpočtu**

Pozorované veličiny jsou vypočítány dle kroků uvedených v předchozí podkapitole. Pro dopočítání hodnot, které se v batygrafických křivkách nebo hydrogramu nenacházejí se využívá metoda lineární interpolace. Tato matematická hodnota využívá nejbližší hodnoty, mezi kterými se interpolovaná hodnota nachází. Metoda zavedená Isaacem Newtonem prokládá dva nejbližší sousední body přímkou. Hledaná hodnota se nachází na této přímce. Funkce pro tuto metodu je následující. [15]

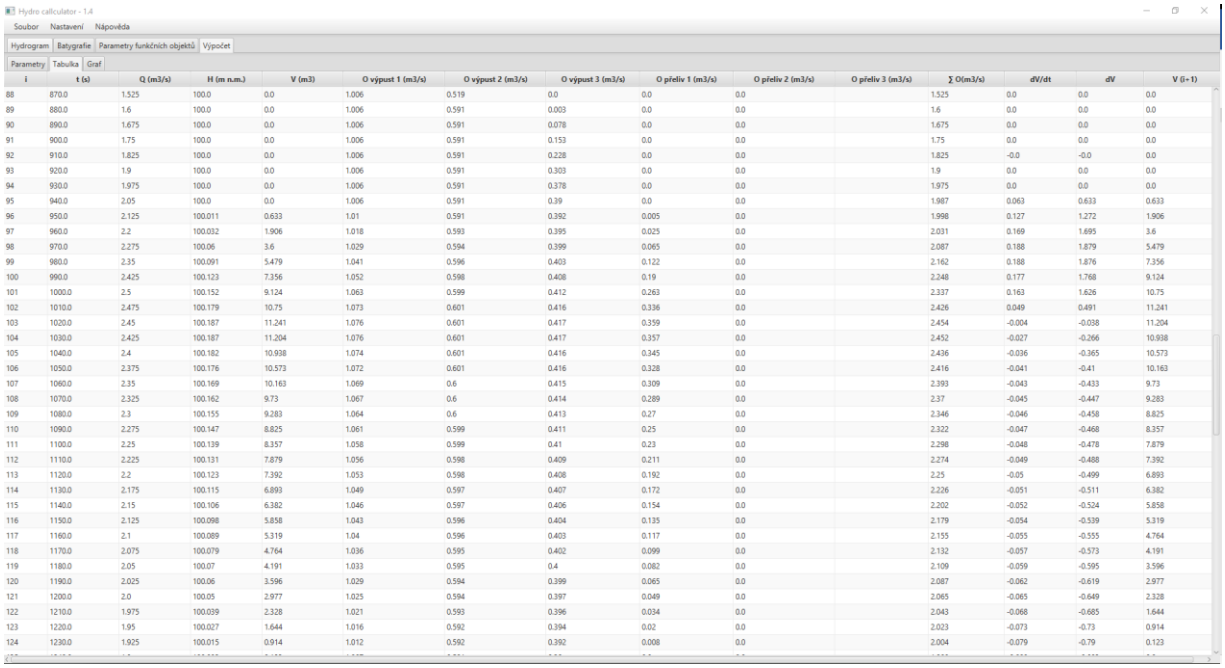
$$\text{Pro } x_0 < x_i < x_1 \; f(x) = f_0 + \frac{f_1 - f_0}{x_1 - x_0}(x - x_0)$$

Dopočítané hodnoty jsou vždy pouze přibližné, skutečná hodnota se může mírně lišit. Z těchto důvodů dochází při výpočtech k různým chybám, které je nutné vyřešit. Mezi tyto chyby patří ošetření záporného objemu vody nebo oscilace výtoků z výpusti. Chyby jsou ošetřovány různými podmínkami. Mezi tyto případy patří například problém s nulovým odtokem z výpusti při nulovém objemu. Vzorec v daném čase totiž nezahrnuje přítok do nádrže, a tedy nedochází k žádnému odtoku. Nicméně reálně tato situace nenastává a odtok je v tomto případě sejný jako přítok.

3.4.3 **Vizualizace dat**

Výsledná vypočtená data jsou vyobrazena ve dvou záložkách. První záložka obsahuje tabulku s patnácti sloupci. V prvním sloupci se nachází index časového kroku. Druhý sloupec obsahuje čas, ve kterém k hodnotám na aktuálním řádku dochází, tento čas je vždy navýšen o stejný časový krok. Ve třetím sloupci zobrazen objem přítoku vody za sekundu v daném čase. Čtvrtý sloupec obsahuje výšku hladiny nádrže. Zde je velmi důležité poznamenat že tato výška je měřena v metrech nad mořem. Pátý sloupec obsahuje hodnotu objemu vody, který se v konkrétním časovém kroku nachází v nádrži. Šestý, sedmý a osmý sloupec obsahuje hodnotu průtoku vody

spodními výpustěmi přehrady. V devátém, desátém a jedenáctém sloupci tabulky se nacházejí průtoky přelivů přehrady. V případě že vodní dílo neobsahuje žádné funkční objekty, tak jsou sloupce s odtoky prázdná. Tímto jsou odlišena od nulových hodnot, které značí že výška hladiny je pod kótu těchto objektů. Ve dvanáctém sloupci se nachází suma všech průtoků. Tato hodnota vyjadřuje objem vody, který v daném čase odteče z nádrže za jednu sekundu. Další sloupec vyjadřuje poměr mezi odtokem a výtokem. Pokud je tento poměr záporný, znamená to, že více vody vytéká, než přitéká. V předposledním kroku je předchozí hodnota vynásobena hodnotou časového kroku. Vyjadřuje tedy množství vody, které přiteče nebo odteče v konkrétním čase. Z této hodnoty je vypočten objem vody v následujícím kroku. Tato hodnota se nachází v posledním sloupci tabulky. Náhled tabulky se nachází na obrázku 8.


Hydro calculator - 1.4

Soubor Nastavení Nápověda

Hydrogram Batygrafie Parametry funkčních objektů Výpočet

Parametry Tabulka Graf

i	t (s)	Q (m3/s)	H (m n.m.)	V (m3)	O výpust 1 (m3/s)	O výpust 2 (m3/s)	O výpust 3 (m3/s)	O přeliv 1 (m3/s)	O přeliv 2 (m3/s)	O přeliv 3 (m3/s)	∑ O(m3/s)	dV/dt	dV	V (i+1)
88	870.0	1.525	100.0	0.0	1.006	0.519	0.0	0.0	0.0		1.525	0.0	0.0	0.0
89	880.0	1.6	100.0	0.0	1.006	0.591	0.003	0.0	0.0		1.6	0.0	0.0	0.0
90	890.0	1.675	100.0	0.0	1.006	0.591	0.078	0.0	0.0		1.675	0.0	0.0	0.0
91	900.0	1.75	100.0	0.0	1.006	0.591	0.153	0.0	0.0		1.75	0.0	0.0	0.0
92	910.0	1.825	100.0	0.0	1.006	0.591	0.228	0.0	0.0		1.825	-0.0	-0.0	0.0
93	920.0	1.9	100.0	0.0	1.006	0.591	0.303	0.0	0.0		1.9	0.0	0.0	0.0
94	930.0	1.975	100.0	0.0	1.006	0.591	0.378	0.0	0.0		1.975	0.0	0.0	0.0
95	940.0	2.05	100.0	0.0	1.006	0.591	0.39	0.0	0.0		1.987	0.063	0.633	0.633
96	950.0	2.125	100.011	0.633	1.01	0.591	0.392	0.005	0.0		1.998	0.127	1.272	1.906
97	960.0	2.2	100.032	1.906	1.018	0.593	0.395	0.025	0.0		2.031	0.169	1.695	3.6
98	970.0	2.275	100.06	3.6	1.029	0.594	0.399	0.065	0.0		2.087	0.188	1.879	5.479
99	980.0	2.35	100.091	5.479	1.041	0.596	0.403	0.122	0.0		2.162	0.188	1.876	7.356
100	990.0	2.425	100.123	7.356	1.052	0.598	0.408	0.19	0.0		2.248	0.177	1.768	9.124
101	1000.0	2.5	100.152	9.124	1.063	0.599	0.412	0.263	0.0		2.337	0.163	1.626	10.75
102	1010.0	2.475	100.179	10.75	1.073	0.601	0.416	0.336	0.0		2.426	0.049	0.491	11.241
103	1020.0	2.45	100.187	11.241	1.076	0.601	0.417	0.359	0.0		2.454	-0.004	-0.038	11.204
104	1030.0	2.425	100.187	11.204	1.076	0.601	0.417	0.357	0.0		2.452	-0.027	-0.266	10.938
105	1040.0	2.4	100.182	10.938	1.074	0.601	0.416	0.345	0.0		2.436	-0.036	-0.365	10.573
106	1050.0	2.375	100.176	10.573	1.072	0.601	0.416	0.328	0.0		2.416	-0.041	-0.41	10.163
107	1060.0	2.35	100.169	10.163	1.069	0.6	0.415	0.309	0.0		2.393	-0.043	-0.433	9.73
108	1070.0	2.325	100.162	9.73	1.067	0.6	0.414	0.289	0.0		2.37	-0.045	-0.447	9.283
109	1080.0	2.3	100.155	9.283	1.064	0.6	0.413	0.27	0.0		2.346	-0.046	-0.458	8.825
110	1090.0	2.275	100.147	8.825	1.061	0.599	0.411	0.25	0.0		2.322	-0.047	-0.468	8.357
111	1100.0	2.25	100.139	8.357	1.058	0.599	0.41	0.23	0.0		2.298	-0.048	-0.478	7.879
112	1110.0	2.225	100.131	7.879	1.056	0.598	0.409	0.211	0.0		2.274	-0.049	-0.488	7.392
113	1120.0	2.2	100.123	7.392	1.053	0.598	0.408	0.192	0.0		2.25	-0.05	-0.499	6.893
114	1130.0	2.175	100.115	6.893	1.049	0.597	0.407	0.172	0.0		2.226	-0.051	-0.511	6.382
115	1140.0	2.15	100.106	6.382	1.046	0.597	0.406	0.154	0.0		2.202	-0.052	-0.524	5.858
116	1150.0	2.125	100.098	5.858	1.043	0.596	0.404	0.135	0.0		2.179	-0.054	-0.539	5.319
117	1160.0	2.1	100.089	5.319	1.04	0.596	0.403	0.117	0.0		2.155	-0.055	-0.555	4.764
118	1170.0	2.075	100.079	4.764	1.036	0.595	0.402	0.099	0.0		2.132	-0.057	-0.573	4.191
119	1180.0	2.05	100.07	4.191	1.033	0.595	0.4	0.082	0.0		2.109	-0.059	-0.595	3.596
120	1190.0	2.025	100.06	3.596	1.029	0.594	0.399	0.065	0.0		2.087	-0.062	-0.619	2.977
121	1200.0	2.0	100.05	2.977	1.025	0.594	0.397	0.049	0.0		2.065	-0.065	-0.649	2.328
122	1210.0	1.975	100.039	2.328	1.021	0.593	0.396	0.034	0.0		2.043	-0.068	-0.685	1.644
123	1220.0	1.95	100.027	1.644	1.016	0.592	0.394	0.02	0.0		2.023	-0.073	-0.73	0.914
124	1230.0	1.925	100.015	0.914	1.012	0.592	0.392	0.008	0.0		2.004	-0.079	-0.79	0.123

Obrázek 9: Tabulka výsledků

Druhá záložka obsahuje dva grafy, kde jsou opět zobrazena výsledná data. První graf obsahuje křivky, na kterých se vizualizují hodnoty průtoků jednotlivými funkčními objekty. Křivky funkčních objektů jsou zobrazeny pouze pokud daná nádrž tyto objekty obsahuje. Dále se zde nachází křivka celkového odtoku a křivka celkového přítoku. Na ose X se tedy nachází průtok v m^3 a na ose Y čas v hodinách. Zde je, z důvodu lepší přehlednosti, rozdíl oproti tabulce, kde je čas uveden v sekundách. Nad grafem je informace o hodně maximálního odtoku, v jakém čase tato hodnota nastává a jaká v tu dobu byla výška hladiny.

Na druhém grafu je zobrazena stav hladiny. Na ose x je opět čas v hodinách a na ose y se nachází výška hladiny v *m n. m.* Nad grafem se nachází informace o maximální výšce hladiny, v jakém čase tato výška nastává, jak velká byla v tu chvíli hodnota přítoku a odtoku. Příklad tohoto grafu se nachází na obrázku (Obrázek 10).

Obrázek 10: Graf výsledných hodnot

4 TECHNOLOGIE A NÁSTROJE

Stručné pojednání technologiích a nástrojích, které byly použity pro návrh a implementaci aplikace.

4.1 Programovací jazyk Java

Základním kamenem aplikace je objektově orientovaný programovací jazyk Java ve verzi osm. Jazyk byl zvolen proto, že požadavky zahrnovaly multiplatformní použití. Tento požadavek programovací jazyk Java splňuje. Aplikace samotná je zpracována pomocí JVM neboli Java virtual machine, který je dostupný na mnoha platformách. JVM je program využívající virtuální stroj k spuštění dalších programů. [11]

4.2 Klientská platforma Java FX

Uživatelské prostředí aplikace je vytvořeno pomocí open source klientské platformy JavaFX, která se považuje za nástupce Java Swing. Platforma je postavena na skriptovacím jazyce FXML zobrazující rozhraní aplikace. FXML poskytuje strukturu pro budovaní uživatelského prostředí a odděluje aplikační logiku od kódu. GUI je v JavaFX možné vytvářet pomocí Scene-Buildru, který umožňuje jednoduchý návrh uživatelského rozhraní. Za pomoci metody „Drag and drop“ je možné přetahovat komponenty na požadovaná místa a SceneBuilder následně generuje FXML kód. Kód je opět zpětně provázaný se SceneBuildrem, takže změny učiněné v kódu se automaticky projevují i grafickém designeru. [13]

4.3 Vývojové prostředí InteliJ IDEA

Pro vytvoření aplikace bylo použito vývojové prostředí InteliJ Idea od firmy JetBrains. První verze byla vydána již v roce 2001 a byla jednou z prvních IDEs s pokročilou navigací v kódu a integrovanou podporou pro refactoringu. Verze použitá pro vývoj aplikace je i 2018.2.4. InteliJ Idea je moderní vývojové prostředí podporující JVM jazyky mezi které patří Java, Scala, Groovy nebo Kotlin. Prostředí je velmi vhodné pro práci s JavaFX, zejména díky již integrované aplikaci SceneBuilder, možnosti jednoduchého generování JUnit testů nebo intuitivní schopností dohledávat knihovny třetích stran pomocí nástroje Maven. [14]

5 POSTUP IMPLEMENTACE APLIKACE

Cílem této kapitoly je popis implementace aplikace pro řešení retenční funkce nádrže. V první části se nachází požadavky na aplikaci dále use case diagram, následuje popis použitého návrhového vzoru a popis samotné implementace.

5.1 Požadavky na aplikaci

Tato kapitola se zabývá požadavky na aplikaci, které byly vymyšleny při konzultacích s odborníkem zabývajícími se problematikou retenčních nádrží.

5.2 Funkční a nefunkční požadavky

V tabulce 1 se nachází výčet funkčních požadavků a v tabulce 2 se nachází výčet nefunkčních požadavků na aplikaci.

Tabulka 1: Funkční požadavky aplikace

ID požadavku	Název	Popis	Priorita
FP1	Import dat hydrogramu	Uživateli je umožněn import dat ze souboru ve formátu CSV nebo txt.	1
FP2	Úprava dat v hydrogramu	Uživateli je umožněno upravovat jednotlivá data hydrogramu.	1
FP3	Vizualizace hydrogramu	Data hydrogramu jsou uživateli zobrazena v podobě grafu. Jsou vypsány maximální hodnoty grafu.	1
FP4	Import dat batygrafie	Uživateli je umožněn import dat ze souboru formátu CSV nebo txt.	1
FP5	Úprava dat v batygrafii	Jednotlivá data z batygrafických křivek je možné upravovat.	1
FP6	Vizualizace batygrafických křivek	Batygrafické křivky jsou zobrazeny ve formě grafů uživateli.	1
FP7	Zadání spodní výpusti	Uživatel zadá parametry spodních výpustí. Je možné zadat až tři takovéto výpusti. Parametr součinitele výtoku je možné zadat ručně nebo dopočítat z dílčích parametrů.	1
FP8	Tabulka s koeficienty	Uživateli je zobrazena tabulka s koeficienty pro různé typy uzávěrů na spodních výpustech.	2
FP9	Zadání přelivu	Uživatel zadá parametry přelivů. Je možné zadat až tři přelivy.	1

FP10	Různé součinitele pro různé hladiny	Pro až tři hladiny vody je možné zadat různé parametry pro výpočet výtoku na základě výšky hladiny.	1
FP11	Výpočty dle více vztahů	Uživatel si pro jednotlivé hladiny může zvolit vztah, dle kterého se bude výtok z výpusti počítat.	1
FP12	Výsledný výpočet	Uživatel zadává časový krok a vstupní hladinu. Je vypočítán přítok, a výtok v jednotlivých časech. Data jsou zobrazena ve formě tabulky	1
FP13	Vizualizace výpočtu	Vypočítaná data jsou zobrazena ve formě grafu. Je zobrazena výsledná hladina, odtok na funkčních objektech, celkový odtok a přítok.	1
FP14	Export dat	Vypočítaná data je možné exportovat do soboru ve formátu CSV.	1
FP15	Uložení nastavení	Aplikace umožňuje uložit nastavení do souboru.	3
FP16	Načtení nastavení	Aplikace umožňuje načtení nastavení ze souboru pro rychlejší zadávání hodnot, optimalizaci stávajících hodnot.	3
FP17	Zobrazení součinitele přepadu	Vybrané vztahy součinitelů přepadu jsou vyobrazeny v grafu.	3

Tabulka 2: Nefunkční požadavky aplikace

ID požadavku	Název	Popis	Priorita
NP1	Aplikace je multiplatformní	Aplikaci je možné spustit na různých operačních systémech	1
NP2	Uživatelský přívětivá	Aplikace je přehledná, všechny parametry a vstupy jsou detailně popsány.	1
NP3	O aplikaci	Aplikace obsahuje informace, odkud jsou brány použité konstanty.	2

5.3 Use case diagram

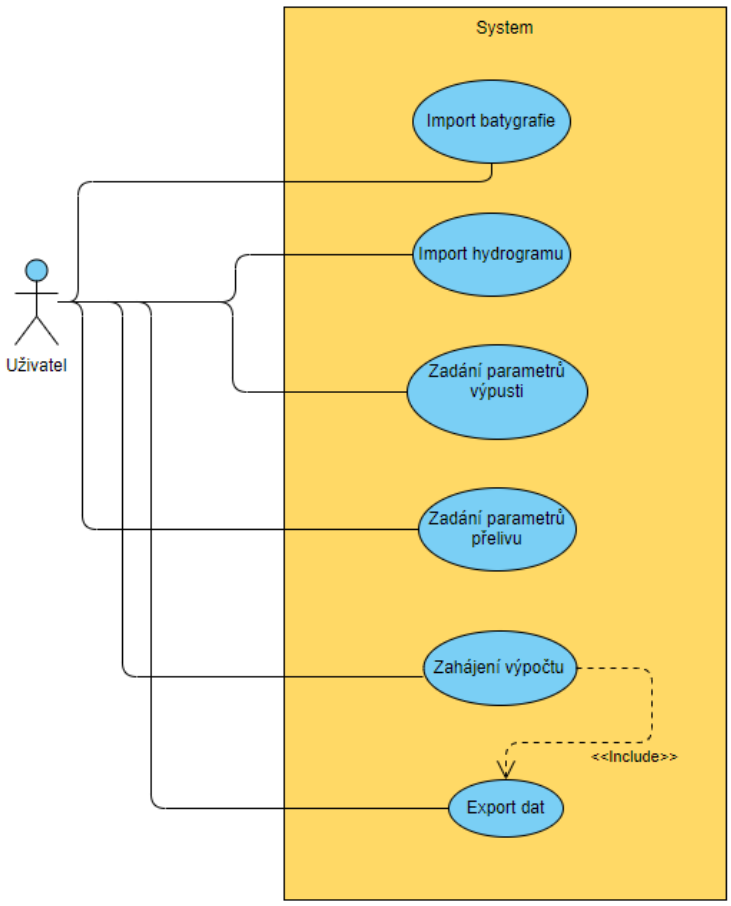
Use case diagram neboli diagram případů užití zachycuje aplikaci tak, jak ji vidí uživatel. Jejím úkolem je grafické znázornění vztahu aktérů a případů užití. [12]

Jediným aktérem je uživatel, který má možnost zadávat hodnoty do systému, tedy spouštět případy užití. Jednotlivé případy jsou následující:

- UC1 Import batygrafie,
- UC2 Import Hydrogramu,
- UC3 Zadání parametrů a počtů spodních výpustí
- UC4 Zadání parametrů a počet přelivů
- UC5 Zahájení výpočtu

- UC6 Export Dat

Diagram případů užití je znázorněn na obrázku 11.



Obrázek 11: Use case diagram

5.4 Architektura aplikace

V této kapitole je popsána architektura, která byla použita pro implementaci softwaru. Program je vybudován jako klientská desktopová aplikace v technologii JavaFx. Pro touto technologii je vhodný architektonický vzor MVC tedy Model-View-Controller. Tento architektonický vzor, jak již vyplývá ze samotného názvu se skládá ze tří hlavních komponent. Jsou to model, view a controller. V následujících odstavcích se nachází jejich detailnější popis a příklad reprezentace komponenty v aplikaci.

Model zpravidla představuje data, jejich zpracování a úpravu. Obsahuje tedy veškerou byznys logiku aplikace. Jsou to nejrůznější servisy pro import a export dat, vzorce pro výpočet, procesory sestavující výpočty a podobně. [12]

Druhou komponentou je view. Jejím úkolem je zobrazit uživateli data, která byla připravena v modelu. U webové aplikace by view byla například klasická HTML stránka. V tomto případě jsou to všechny soubory s příponou „*.fxml*“ obsahující XML objekty, podle kterých je sestaveno uživatelské rozhraní aplikace.

Poslední zmíněnou komponentou je controller. Za úkol má předávat akce uživatele na veiw. Typickým příkladem je stisknutí tlačítka, zmáčknutí klávesy při zadávání do textového pole nebo výběr v combo boxu. Controller následně zpracovává tyto akce a jejich výsledek odesílá do modelu.

Základním principem MVC je tedy oddělit byznys logiku do uživatelského rozhraní. Tím je dosaženo jednodušší modifikace kódu v různých částech aplikace. Například pokud by uživatel chtěl změnit jednotlivé části grafického rozhraní, není nutné zasahovat do tříd kde se nachází kód s výpočty a naopak. [12]

Návrhový vzor MVC má více variant, které je možné uplatnit. Za zmínku stojí MVP neboli model-view-presenter či model-delegate. Od sebe se liší zejména komunikací view a controlleru. Komponenta model je vždy oddělena. V této aplikaci je použit převážně vzor MVP, protože u čistého MVC by například komponenty pro vstup měli být odděleny od komponent pro prezentaci. [12]

5.5 Uživatelské rozhraní

Uživatelské rozhraní je definováno soubory s koncovkou *fxml*. Nachází se v balíčku *views*. Aplikace je rozdělena na sedm hlavních záložek a každá záložka má vlastní view. Tato dílčí view jsou obsažena v komponentě *TabPane* zobrazující se jako záložka, v hlavním view nazvaném *mainView*. Jména dílčích view jsou odvozena od toho, co mají za úkol zobrazit. Nesou tedy názvy *hydrogramView, batygrafiView, functionObjectView, outletView* a *overflowView*. Poslední dvě zmíněná view jsou umístěny ve *functionObjectView,* důvodem je, že v aplikaci je použito více instancí těchto komponent. Konkrétně jsou to tři výpusti a tři přelivy. FXML kód použitý pro vložení dílčích view do *mainView* je následují.

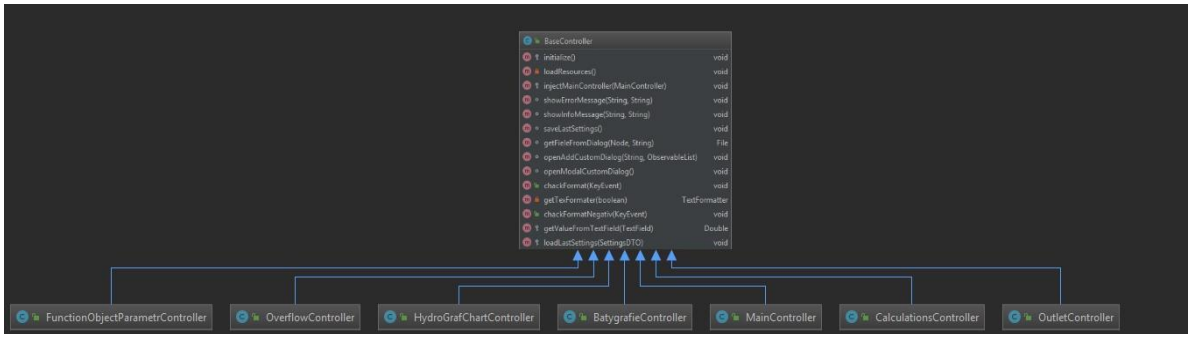
```
<TabPane prefHeight="289.0" prefWidth="500.0" tabClosingPolicy="UNAVAI-
LABLE" BorderPane.alignment="CENTER">
            <tabs>
                <Tab text="%key.hydrograf">
                    <content>
                        <fx:include fx:id="hydroChart"
                         source="hydroGramView.fxml" />
                    </content>
                </Tab>
                <Tab text="%key.batygrafie">
                    <content>
```

```
                        <fx:include fx:id="batygrafie"
                         source="batygrafiView.fxml" />
                    </content>
                </Tab>
                <Tab text="%key.FunctionObjectParametr">
                    <content>
                        <fx:include fx:id="functionObjectParametr"
                         source="functionObjectView.fxml" />
                    </content>
                </Tab>
                <Tab text="%key.callculation">
                    <content>
                        <fx:include fx:id="calcullation"
                         source="fallculationView.fxml" />
                    </content>
                </Tab>
            </tabs>
        </TabPane>
```

Každé view má vlastní controller, který se nachází v balíčku *contoller*. Třídy typu controller jsou potomkem třídy *BaseController*, která obsahuje společné části kódu. Jsou to například metody pro vyvolání dialogových oken či metody pro validace vstupů nebo inicializace třídy s textovými zdroji. Diagram hlavních controllerů se nachází na obrázku 12. Na tomto diagramu jsou pro lepší přehlednost skryty názvy metod, kterých Controllery obsahují spousty. Jejich úkolem je odchytávat action eventy pro obsluhu komponent. Dále například vykreslující grafy. Mezi metody pro vykreslení grafu patří i následující.

```
Public void updateWaterLevlChartSeries(String outletName, List<Float> ti-
meStep, List<Float> waterLevl) {

        Float max = getMaximum(waterLevl);
        if (max != null && maxYWaterLevl < max) {
            maxYWaterLevl = max;
            NumberAxis yAxisLocal = ((NumberAxis)
            waterLevlChart.getYAxis());
            yAxisLocal.setAutoRanging(false);
            yAxisLocal.setUpperBound((int) maxYWaterLevl + 1);
        }

        Float min = getMinimum(waterLevl);
        if (min != null && minYWaterLevl > min && min > 0) {
            minYWaterLevl = min;
            NumberAxis yAxisLocal = ((NumberAxis)
            waterLevlChart.getYAxis());
            yAxisLocal.setAutoRanging(false);
            yAxisLocal.setLowerBound(minYWaterLevl);
        }


        ChartUtils  chartUtils = new ChartUtils();
        ObservableList<XYChart.Data<Float, Float>> data =
        chartUtils.createChartDataFloat(
        chartUtils.createBackingDataFloat(timeSteps, waterLevl));
        Series<Float, Float> serie = new Series(outletName, data);
        waterLevlChart.getData().add(serie);
    }
```

Hlavní controller se nazývá *MainController* a drží instance jednotlivých dílčích tříd typu controller. Tyto controllery obsahují referenci na tento hlavní controller, čímž je zajištěna oboustranná komunikace.



Obrázek 12: Graf Controllerů

V aplikaci jsou využívána dialogová okna, pro různou interakci s uživatelem. Tato dialogová okna obsahují opět vlastní view a controller. Nachází se v balíčku *dialogs.view* případně *dialogs.controller*. Tato dílčí dialogová okna by bylo pro zjednodušení možné napsat v klasickém Java kódu ale pro zajištění a architektury je použit stejný princip jako při tvorbě hlavních oken aplikace. Controllery těchto dialogových oken jsou potomky třídy BaseCustomAddDialogController.který je opět potomkem třídy *BaseController*,

Posledním druhem veiw a controller jsou detaily jednotlivých vztahů, podle kterých se počítají odtoky přelivů. Jednotlivé detaily se vzájemně hodně liší a pokud by byly umístěny hlavním controlleru stala by se tato třída velmi nepřehlednou. Proto vztahy dostaly vlastní view a controller. Nachází se v balíčku v *details.view* případně *details.controller*. Tyto controllery mají opět vlastního předka *DetailBaseController*. Oboustranná komunikace je opět zajištěna takzvanou injection neboli vložením controlleru, který instance detailů vytváří. V praxi to znamená, že controller s detailem obsahuje referenci na controller, který tuto instanci vytvořil. Jejich úkolem je vytvořit DTO objekt popisující způsob výpočtu. Popis DTO objektů bude vysvětlen následujících kapitolách.

5.6 Byznys logika aplikace

Byznys logika je oddělena za pomoci návrhového vzoru MVC od uživatelského prostředí. V tomto případě do byznys logiky patří čtyři hlavní balíčky *dto, models, processors,* a *services.*

V balíčku *dto* se nachází takzvané Data-Transfer-Object. Tyto objekty mají za úkol přenášet data do jednotlivých vrstev aplikace. Jsou to například vstupy od uživatele transformované do DTO objektu na controlleru. Tato data je následně mnohem snažíš odeslat do vyšší vrstvy aplikace, kde se s nimi dále pracuje. Třídy nacházející se v balíčku *dto* tedy obsahují převážně své atributy a metody *get()* a *set().* Výhodou těchto objektů je například i jejich jednoduché zobrazení v různých komponent typu table view, které umožňuje i mapování na základně názvu atributů objektu.

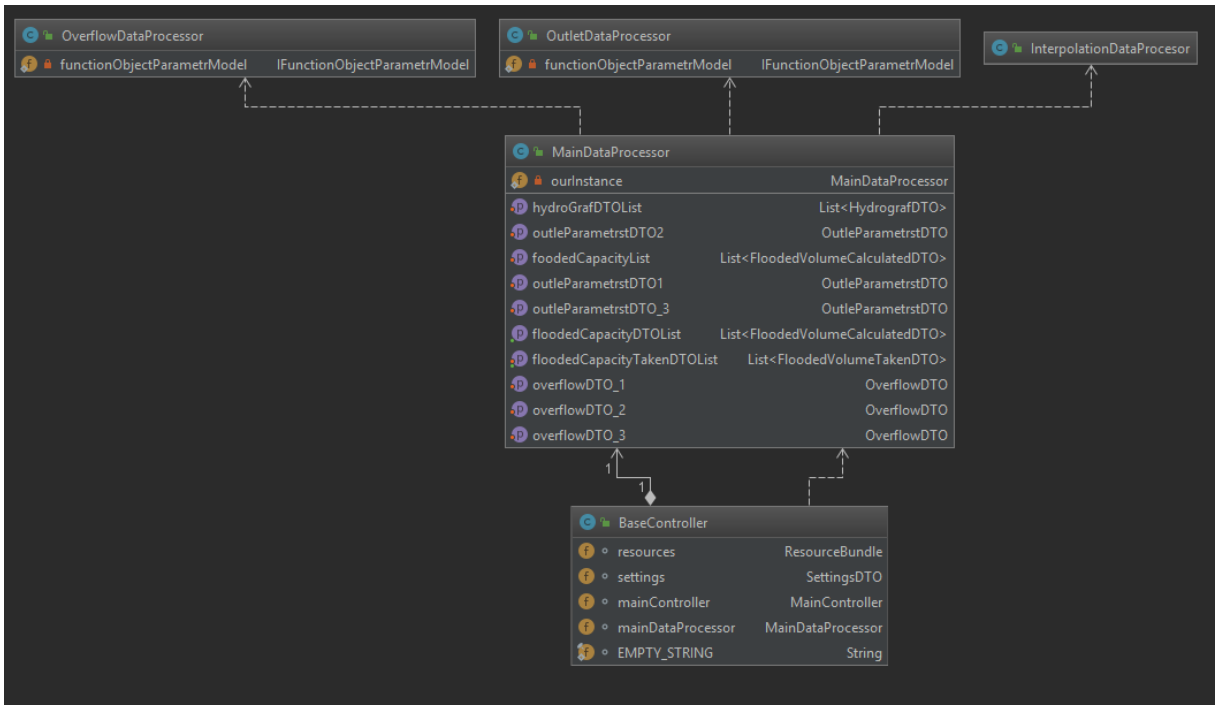
Dalším balíčkem je balíček *models.* Jsou zde třídy *BatygrafieModel, HydrogramModel* a *FunctionObjectModel*. Tyto třídy obsahují veškerou logiku pro práci s jednotlivými částmi aplikace. Nacházejí se zde například metody, které se starají o výpočet dílčích vzorců. Metody naplňující DTO objekty pro tabulky a podobně. Každá z těchto metod má svůj vlastní interface, aby bolo možné tyto třídy testovat či lépe upravovat jejich logiku. Níže je uveden příklad metody vypočítávající součinitel přepadu dle Smetany.

```
public double calculateRelationsOverflowSmetana(double h, double hm){
    if(hm == 0)
        return 0;
    return 0.499 * (0.63+0.37 * Math.sqrt(h/hm));
}
```

Předposledním balíčkem, který je pro porozumění aplikaci potřeba popsat je balíček processors. Nejdůležitější třídou je zde MainDataProcessor. Je navržena pomocí návrhového vzoru singelton. V životním cyklu aplikace existuje tedy její jediná instance, která je vytvořena ve třídě *BaseController*. Tímto se tato jediná instance stává přístupnou všem controllerům a umožňuje jim posílat do této třídy svá data. Třída *MainDataProcessor* uchovává tato data ve formě DTO objektů a zajišťuje jejich zpracování. Nachází se zde i metoda tvořící základ pro výstupní výpočty. Jejím úkolem je volání metod ostatních tříd v tomto balíčku, kde jsou počítána data pro dílčí části programu. Příkladem může být třída *OutletDataProcesor* počítající data pro spodní výpusti nebo *OverflowDataProcessor*, kde probíhají výpočty dat přepadů či *InterpolationDataProcessor.* Poslední zmíněná třída obsahuje metody pro dopočítání dat za pomoci matematicko statistické metody lineární interpolace. Příklad kódu pro výpočet je následující.

```
static double GetWaterLevlByVolume(List<FloodedVolumeCalculatedDTO> floo-
dedCapacityDTOList, double volume) throws Exception {
        floodedCapacityDTOList.sort(Comparator.comparingDouble(FloodedVo-
lumeCalculatedDTO::getBpv).reversed());

        FloodedVolumeCalculatedDTO minValue;
        FloodedVolumeCalculatedDTO nexValue;

        Optional<FloodedVolumeCalculatedDTO> cleanValue =
          floodedCapacityDTOList.stream()
          .filter(x -> x.getVolume() == volume).findFirst();
        if(cleanValue.isPresent())
            return cleanValue.get().getBpv();

        Optional<FloodedVolumeCalculatedDTO> minCapacity =
          getMin(floodedCapacityDTOList,
          Comparator.comparing(FloodedVolumeCalculatedDTO::getVolume));
          Optional<FloodedVolumeCalculatedDTO> maxCapacity =
            getMax(floodedCapacityDTOList,
            Comparator.comparing(FloodedVolumeCalculatedDTO::getVolume));

        if(!minCapacity.isPresent() || !maxCapacity.isPresent())
            return 0;

        if(minCapacity.get().getVolume() > volume){
            DecimalFormat decimalFormat =
             DPNumberFormater.getDecimalFormater();
            throw new Exception(MessageFormat.format(
             Messages.getMessage("EX05"), decimalFormat.format(volume)));
        }else if(maxCapacity.get().getVolume() < volume){
            DecimalFormat decimalFormat =
             DPNumberFormater.getDecimalFormater();
            throw new Exception(MessageFormat.format(
             Messages.getMessage("EX06"), decimalFormat.format(volume)));
        }else{
            minValue = floodedCapacityDTOList.stream()
             .filter(x -> x.getVolume() <= volume).findFirst().get();
            nexValue = floodedCapacityDTOList
             .get(floodedCapacityDTOList.indexOf(minValue)-1);
        }

        return minValue.getBpv() + (volume - minValue.getVolume()) *
         ((nexValue.getBpv() - minValue.getBpv()) / (nexValue.getVolume()
         - minValue.getVolume()));
    }
```

Dílčí procesory drží instance tříd z balíčku *models*, ve kterých jsou mimo jiné i metody pro výpočet jednotlivých vzorců uvedených v kapitole 4. Diagram balíčku *processors* je vyobrazen na obrázku 10.



Obrázek 13: Diagram balíčku Processors

Poslední hlavní balíček byznys logiky nese název *services*. V této sekci jsou umístěny implementace služeb pro export a import dat. Čtení textových soborů, nebo uložení nastavení aplikace. Jsou to následující třídy *DataCVSExportProcessorService*, která má za úkol připravit data pro export do textového souboru. Třída *DataCVSExportService*, jejímž úkolem je zapisovat do textových souborů ve formátu CSV. Dále se zde nachází *DataTextProcessorService* zpracovávající importované textové soubory a *FileReaderService*, která má za úkol čtení textových souborů. Poslední třídou v tomto balíčku je *SettingService* zprostředkující ukládání nastavení aplikace. Pro uložení nastavení využívá knihovnu Jackson 2, která umožňuje jednoduše serializovat objekt ve formátu Json do souboru a opět tento soubor deserializovat zpět. Při práci z toho knihovnou je nutné, aby všechny objekty, které knihovna sterilizuje měly výchozí konstruktor. Dále pokud je zapotřebí sterilizovat objekt typu list. Musí být tato třída konkrétního datového typu. Knihova je dostupná jako opensource a je možné stáhnout pomocí nástroje Maven nebo z webové služby GitHub. Kód, který názorně předvádí práci s touto knihovnou je uveden na následující straně.

```
public static void saveAppSettings(SettingsDTO settings){

      ObjectMapper mapper = new ObjectMapper();

      try {
          mapper.writeValue(new File("/settings.json"), settings);
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
    }


     public static SettingsDTO loadAppSettings() throws IOException {
        ObjectMapper mapper = new ObjectMapper();
        return mapper.readValue(new File("/settings.json"),
        SettingsDTO.class);
    }
```

Dále aplikace obsahuje balíček se zdroji s názvem *resources*. Zde jsou uloženy obrázky a texty do aplikace. Je vhodné zmínit, že aplikace obsahuje balíček *lang.*properties může obsahovat více jazykových sad. Tato vlastnost tedy umožňuje snadnou lokalizovat do jiného jazyka. Aby bylo možné s těmito zdroji pracovat i v jiných vrstvách softwaru, nachází se zde statická třída *Messages,* která je inicializována ve třídě *BaseController* a obsahuje veškeré texty ze souboru *lang.properties.* Třída obsahuje metodu *getMessage*, kde je možné za pomoci klíče jednoduše získat požadovaný textový řetězec. Příklad volání této metody je následující.

```
String message = Messages.getMessage("EX05");
```

V projektu se dále nachází balíček *helprs, enums, tests*. Balíček *helprs* obsahuje třídy *ChartUtils,Components, DPNumberFormater, UTF8PropertisProcessor* a *TestData*. Prní třída má za úkol zpracovávat jednotlivé komponenty grafů. *DPNumberFormater* slouží k sjednocení formátování čísel napříč aplikací. *UTF8PropertisProcessor* načítá data z *lang.properties* ve formátu UT8, nativně je toto formátování podporováno až od Java 9. Třída *TestData* obsahuje hodnoty testovací hladiny, které jsou využity při kreslení grafů pro vztahy součinitelů přepadu. Tento balíček obsahuje v podstatě kód používaný primárně v controllerech, který nebylo možná dát do předka a je potřeba na více místech. Sekundárním úkolem těchto tříd je zabránění zbytečné duplicity kódu.

Dalším zmíněným balíčkem je balíček *enums*. Ten, jak již vypovídá z názvu, obsahuje třídy typu enum tedy třídy výčtových typů. Názvy těchto tříd jsou *BatygrafyType, RelationOverflowType* a *ZachovalAlternativeType.*

Balíček *tests* obsahující JUnit testy pro kontrolu správnosti výpočtů. Tento balíček byl vytvořen jako poslední, nicméně stal se jedním z nejdůležitějších balíčků při konečném dolaďování aplikace. Nebýt těchto testovacích metod ověřování správnosti výpočtů by bylo mnohem náročnější. Příklad kódu JUnit testu se nachází na další stránce.

```
FunctionObjectParametrModel fu = new FunctionObjectParametrModel();
    private double H = 0.4;
    private double H1 = 1;
    private double t = 0.9;
    private double P = 1.2;
    private double i = -0.01;
    private double r = 0.1;
    private double so = 0.066;
    private double ks = 0.05;

    @Test
    public void processCalculationOwerflow() {
        double res =  fu.processCalculationOwerflow(0.36,2,0.472);
        DecimalFormat decimalFormat =
        DPNumberFormater.getDecimalFormater();
        res = Double.valueOf(decimalFormat.format(res));

        assertEquals(res, 1.034, 0.05);
    }

    @Test
    public void callculateOWZachovalCpH() {
        double result = fu.callculateOWZachovalCp(H , P);
        assertEquals(result, 1, 0.05);
    }

    @Test
    public void callculateOWZachovalCpH1() {
        double result = fu.callculateOWZachovalCp(H1 , P);
        assertEquals(result, 1.02, 0.02);
    }

    @Test
    public void callculateOWZachovalCtH1() {
        double result = fu.calculateOWZachovalCt(H1 , t);
        assertEquals(result, 1.17, 0.01);
    }

    @Test
    public void callculateOWZachovalCKsH1() {
        double result = fu.callculateOWZachovalCks(H1, t, ks);
        assertEquals(result, 0.99, 0.01);
    }
```

6 EXPERIMENTY

Za pomocí aplikace byla provedena sada experimentů na vybrané retenční nádrži. Data z toho experimentu by bylo možné využít například při renovaci či rozšíření nádrže nebo vylepšení její bezpečnosti. Výstupem první části experimentu, je zjistit stavy nádrže při aktuálních parametrech funkčních objektů. V druhé části experimentu jsou vyhledávány optimální šířky bezpečnostních objektů. Úkolem třetí části experimentu je zjistit, zda by bylo možné, aby výtok spodní výpustí byl dostačující pro ochranu hráze před přelitím.

6.1 Popis nádrže

Nádrž, na níž je experiment prováděn má sypanou hráz. Součástí hráze je objekt spodní výpusti a boční nehrazený přeliv. V čase 0 je výška hladiny pod úrovní maximálního zásobního prostoru. 315,5 m. n.m. Ve výšce 316,4 m. n.m. je umístěn bezpečnostní přeliv. Koruna hráze je umístěna v 319 m. n.m. Maximální hladina 317,9 m. n.m.

6.2 Vstupní parametry

Parametry, potřebné k dosažení experimentu. Pro vizualizaci těchto parametrů jsou použity snímky z aplikace, jež byla pro tyto potřeby vytvořena.

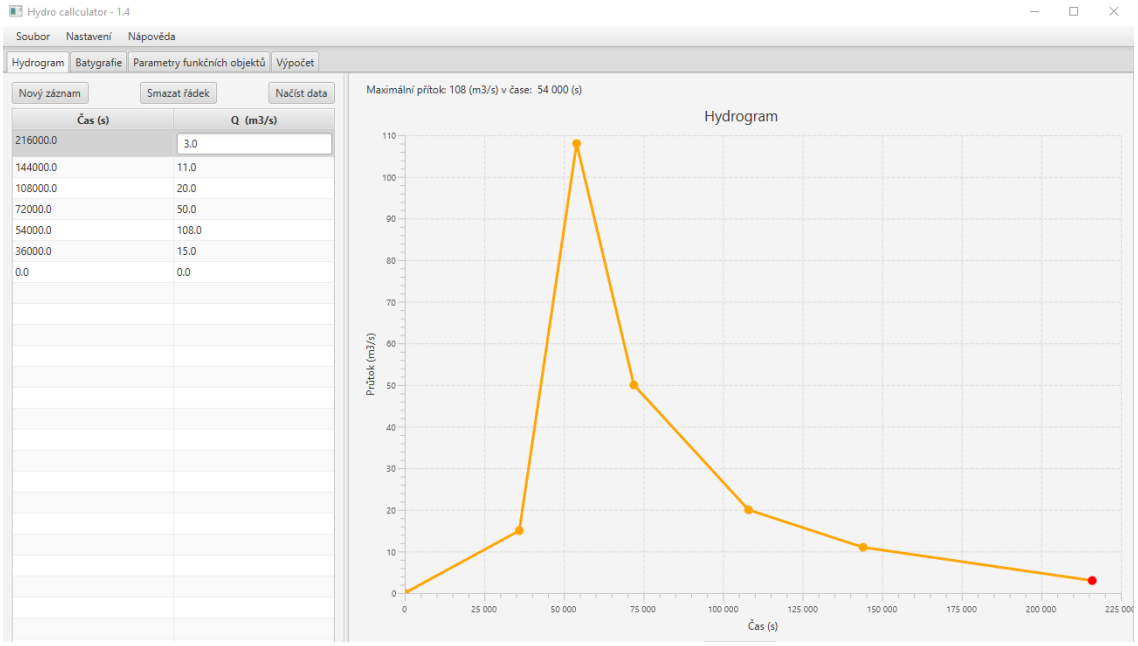
6.2.1 Hydrogram

Hydrogram, na kterém je experiment prováděn odpovídá hodnotám z tabulky 3. Nachází se zde poměrně vysoké hodnoty v relativně krátkém čase. Tyto případy nastávají v posledních letech i na našem území. Kdy na místo dlouhých mírných a vytrvalých dešťů, přicházejí krátkodobé intenzivní srážky. Proto v době největší intenzity deště hydrogram velice rychle roste, a vytvoří objemnou povodňovou vlnu.

Tabulka 3: Data pro hydrogram experimentu

Čas (s)	Q (m^3/s)
216000	3
144000	11
108000	20
72000	50
54000	108
36000	15
0	0

Náhled povodňové vlny pro tento hydrogram je možné vidět na obrázku 14.



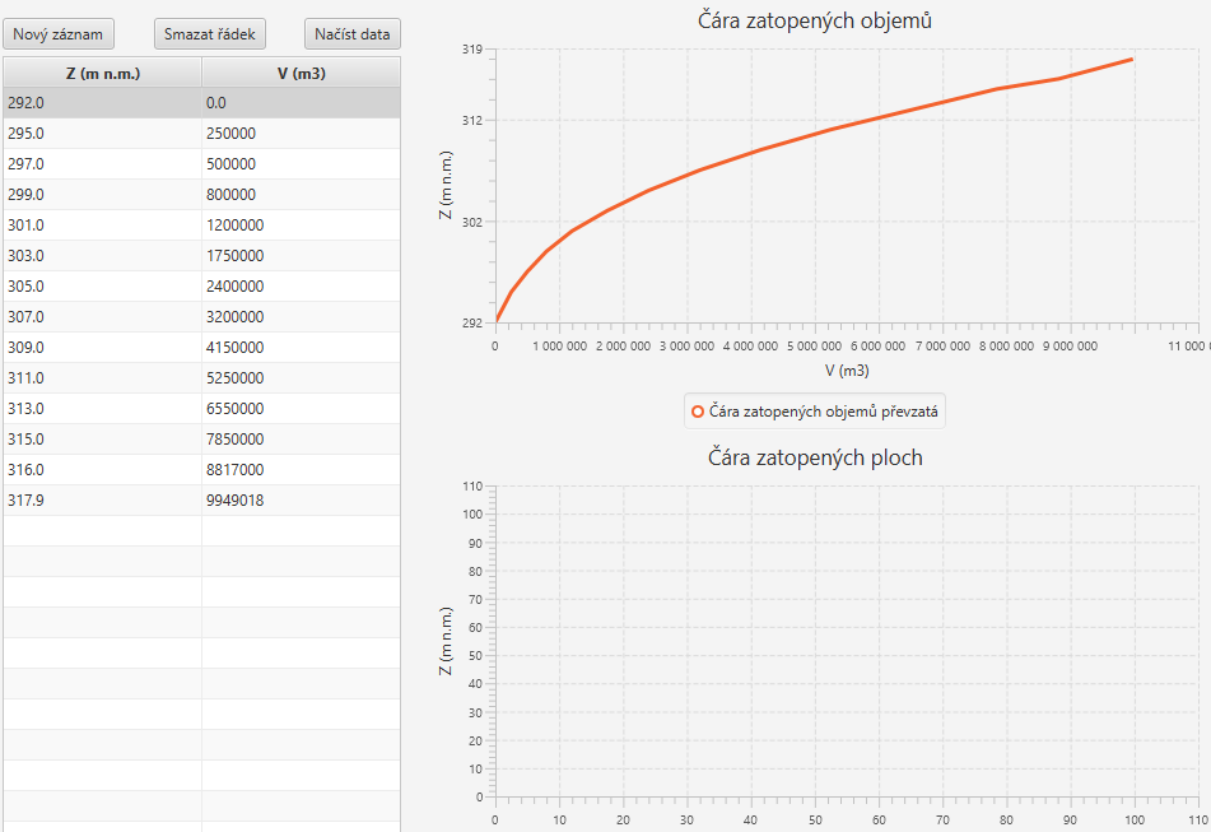
Obrázek 14: Data pro hydrogram experimentu

6.2.2 **Batygrafické křivky**

Batygrafické data přehrady odpovídají datům z tabulky 4. Při výšce hladiny 229 m n.m. je přehrada prázdná. Tento objem může nastat jen při nulovém přítoku. Maximální výška hladiny je 317,9 metru. Při překročení tého hodnoty dochází k přelivu vody přes hráz přehrady. Na obrázku 15 je vidět čára zatopených objemům, která vizualizuje data z batygrafie a popisuje tvar a kapacitu retenční nádrže.

Tabulka 4: Data batygrafické křivky experimentu

Z (m n.m.)	V(m^3)
229	0
295	250000
297	500000
299	800000
301	1200000
303	1750000
305	2400000
307	3200000
309	3110000
313	6550000
315	7850000
316	8817000
317.9	9949018



Obrázek 15: Batygrafická křivka experimentu

6.2.3 **Spodní výpusť**

Nádrž experimentu má pouze jednu spodní výpusť o průměru potrubí 0,8 m, a je umístěna v 288,1 m n. m. Součinitel výtoku μ má hodnotu 0,416.

6.2.4 **Bezpečnostní přeliv**

Přeliv nádrže je umístěn ve výšce 316,4 m. n.m. Jeho současná šířka je 26 m. Součinitel v přepadu je zadán manuálně a jeho hodnota je 0,51.

6.2.5 **Parametry výpočtu**

Výchozí hladiny v čase 0 je 315,5 m. n.m. Časový krok pro experiment je stanoven vždy na 10 s.

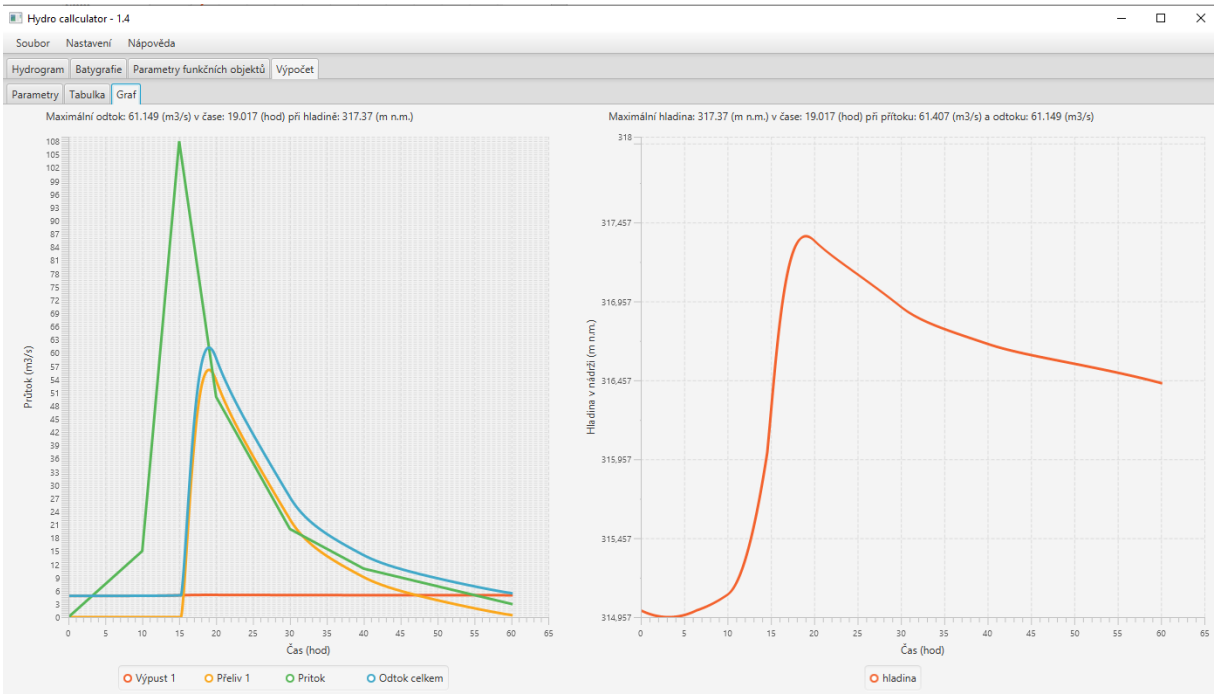
6.3 Výsledky experimentů

Výsledky provedených experimentů jsou zobrazeny na obrázcích z aplikace, či zpracovány pomocí tabulkového editoru Microsoft Excel.

6.3.1 Současný stav nádrže

Na obrázku 16. je vidět jaké hodnoty vykazují pozorované veličiny na hrázi v současném stavu. Výpusť má konstantní průtok kolem 5 m^3/s, toto je dáno průměrem výpustného zařízení.

Do 15 hodin od času 0 se hráz plní. Po patnácté hodině, jak je vidět v pravém grafu obrázku 16., se výška hladiny dostává na úroveň kóty bezpečnostního přelivu a dochází k průtoku tímto bezpečnostním objektem. Maximální průtok na tomto objektu je 60 m^3/s. Z obrázku je patrné že objekt zvládá odvádět vodu z povodňové vlny a výška hladiny nikdy nepřekročí 317,4 m. n.m.



Obrázek 16: Výsledek experimentu současného stavu nádrže

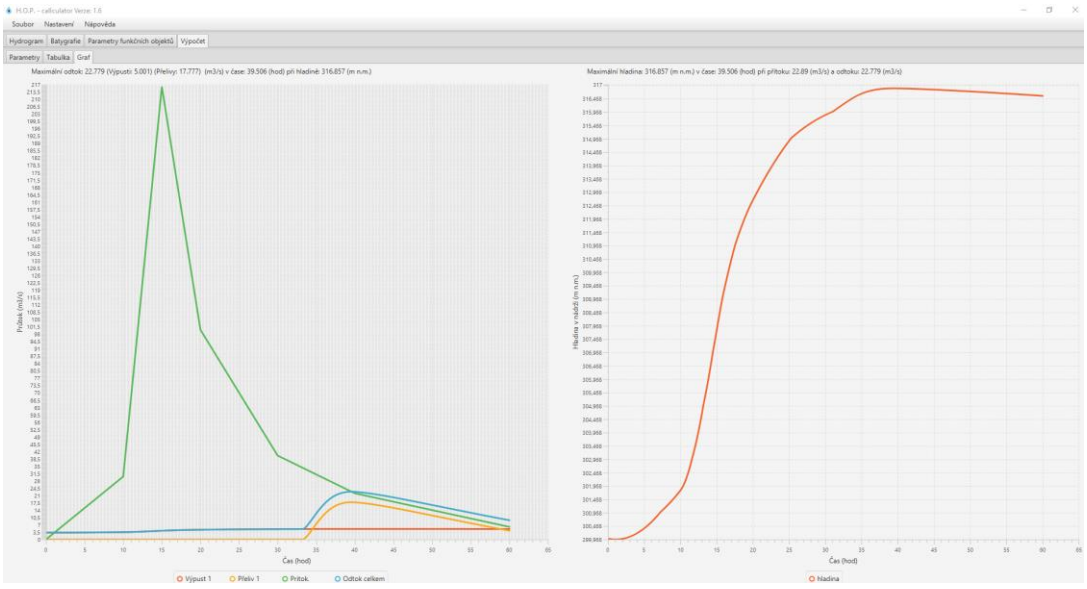
6.3.2 **Stav nádrže při znásobeném přítoku**

Tento experiment simuluje stav nádrže při povodňové vlně, která nabývá dvakrát větších hodnot než povodňová vlna předchozí. Tento stav by mohl nastat v současné době pouze za extrémních podmínek. Hodnoty pro hydrogram jsou uvedeny v tabulce 6.

Z obrázku 17 je patrné i že i při takto velkém objemu přítoku. Je přeliv stále schopen odvádět vodu z přehrady, aniž by došlo k jejímu přetečení. K přetečení dochází až v případě, že by tyto zvětšené hodnoty byly ještě jednou zdvojnásobeny.

Tabulka 5: Hodnoty znásobeného hydrogramu druhého experimentu

Čas (s)	Q (m^3/s)
216000	6
144000	22
108000	40
72000	100
54000	216
36000	30
0	0

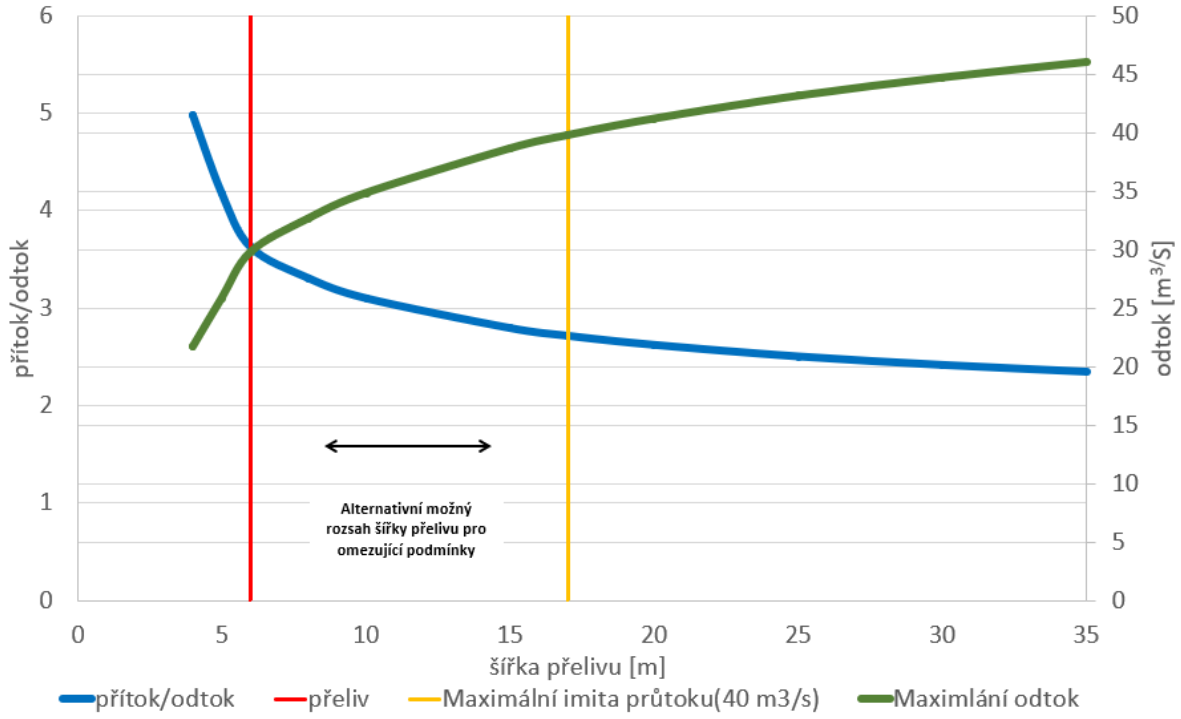


Obrázek 17: Stav přehrady při extrémních hodnotách hydrogramu

6.3.3 Vhodná šířka přelivu

Experiment prověřuje různé variace šířky přelivu. Úkolem je najít takovou hodnotu, kdy by nedošlo k přetečení nádrže a zároveň celkový odtok by byl menší než 40 m^3/s. Pokud by byl odtok vyšší, než toto množství docházelo by již k zatopení oblastí pod nádrží. Výsledky experimentu by bylo možné použít pro případnou rekonstrukci přelivu. Rozmezí bylo stanoveno od 4 do 35 metrů.

Výsledky tohoto experimentu jsou vidět na obrázku 17. Z těchto výsledů vyplývá že vyhovující šířkou přelivu pro povodňovou vlnu experimentu je 6–17 m.

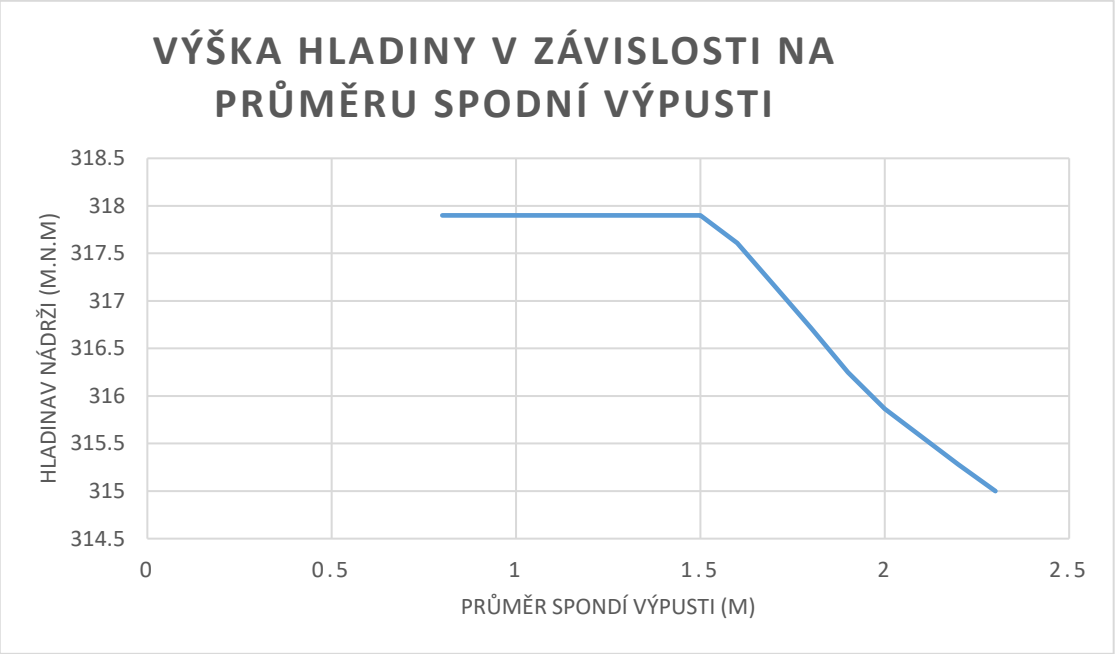


Obrázek 18: Graf závislosti výšky hladiny a šířky přelivu

6.3.4 Maximální výška hladiny v závislosti na šířce spodní výpusti

Další experiment sleduje, jak se mění maximální výška hladiny v závislosti na šířce spodní výpusti. Cílem je nalézt šířku výpusti, která by zvládla převést povodňovou vlnu z hydrogramu experimentu, a nebylo nutné chránit hráz před přelitím pomocí přelivu. Pro tento experiment

bylo stanoveno rozmezí šířky přelivu od 0,8 m do 2,3 m. Z obrázku 18. vyplývá, že pro zvládnutí takovéhoto přítoku by výpusť musela mít minimálně 1,6 m.



Obrázek 19: Graf závislosti výšky hladiny a průměru spodní výpusti

6.4 Závěr experimentů

Z výsledků pozorování vyplývá že, nádrž, na níž byly experimenty uskutečněny zvládá zcela transformovat povodňovou vlnu danou výše uvedeným hydrogramem.

Změna klimatu mění i charakter srážek, které se promítají i do změn objemu přítoku povodňové vlny. Z těchto důvodu může v budoucích letech nastat situace, že hodnoty uvedené v hydrogramu budou i mnohem větší. Na tyto hodnoty, jak vyplývá z druhého experimentu je opět nádrž velice dobře připravena, a není nutná žádná rozsáhlá rekonstrukce. Pouze by bylo vhodné bezpečnostní přeliv osadit konstrukcí, která umožní jeho zúžení a tím se sníží maximální průtok pod 40 m^3/s. Nahrazení bezpečnostního přelivu rozšířením spodní výpustí, se také nejeví jako vhodné. Důvodem je zejména to, že šířka potrubí by se musela zdvojnásobit a tím by při rekonstrukci došlo k velkému zásahu do sypané hráze a mohlo by dojít k narušení její struktury.

ZÁVĚR

Úkolem mé diplomové práce bylo navrhnout a implementovat softwarový nástroj, který umožní realizaci vybraných matematických přístupů pro řešení bilanční rovnice přítoku, odtoku a retence vody v nádrži.

Prvním krokem bylo seznámení se s touto problematikou, které probíhalo na konzultacích s odborníkem z praxe a rešerší aktuálně používaných nástrojů. V tomto stádiu bylo nejdůležitější a zároveň i nejtěžší najít společné porozumění při tvorbě požadavků na aplikaci. Dále bylo nutné pochopit základní principy oborou, o němž jsem doposud neměl žádné povědomí. Nejvíce vědomostí jsem získal právě z konzultací. Zde jsem nabyl i spoustu dalších zkušeností, které věřím že využiji i v budoucnu zejména při komunikaci se zákazníkem. Například často bylo třeba rozlišovat, které z poskytnutých informací jsou pro mě relevantní a které ne.

Aplikaci jsem budoval postupně ve formě modulů. Nejdříve jsem vytvořil část, kde byl pouze hydrogram, dále jsem vytvářel modul s batygrafií až jsem se propracoval k finálním výpočtům. Tímto principem došlo k usnadnění tvorby uživatelského rozhraní, které nebylo nutné vytvářet předem pro celou aplikaci a mohl jsem se zaměřit pouze na jednotlivé části. Vytvořená část následně byla konzultována s odborníkem, který bude software využívat. Na stejné oddíly jsou rozděleny i postupy výpočtů uvedené v teoretické části. Tato část dále obsahuje všechny podrobné specifikace aplikace, jako jsou použité technologie nebo popis vlastní implementace. Také se zde nachází experiment provedený za pomoci nově vytvořeného softwaru, který ověřuje bezpečnost retenční nádrže.

Praktická část v tuto chvíli splňuje všechny funkční i nefunkční požadavky stanovené při návrhu nebo ve stádiu vývoje aplikace. Nicméně bylo by ji možné ještě rozvinout. Rád bych například přidal možnost další jazykovou lokalizaci, na což je nástroj připraven. Dále by bylo vhodné přidat možnosti importu dat i v jiných jednotkách. Upravit design aplikace tak aby odpovídal modernímu prostředí. Nebo sloučit výstupní grafy do jednoho přehledného.

Navržený softwarový nástroj byl následně využit pro simulační experimenty, jež jsou součástí odborné publikace, která byla v čase odevzdání této diplomové práce v recenzním řízení na mezinárodní simulační konferenci EMSS 2019 a jejíž jsem spoluautorem.

POUŽITÁ LITERATURA

[1] KRATOCHVÍL, Jiří, Miloš JANDA a Vlastimil STARA. *Projektování přehrad: komplexní projekt HT*. Brno: Ediční středisko Vysokého učení technického, 1988.

[2] HAVLÍK, Aleš. *Nádrže a přehrady* [online]. In:. 2007, s. 57 [cit. 2019-03-27]. Dostupné z: http://hydraulika.fsv.cvut.cz/Vin/ke_stazeni/Nadrze_prehrady.pdf

[3] VRÁNA, Karel a Václav DAVID. *Bezpečnostní přelivy* [online]. In: ČVUT v Praze – Fakulta stavební Katedra hydromeliorací a krajinného inženýrství, 2018, s. 28 [cit. 2019-03-27]. Dostupné z: http://storm.fsv.cvut.cz/data/files/p%C5%99edm%C4%9Bty/VK2/VK2_predn06.pdf.

[4] VRÁNA, Karel a Václav DAVID. *Funkční objekty* [online]. In:. ČVUT v Praze – Fakulta stavební Katedra hydromeliorací a krajinného inženýrství, 2018, s. 28 [cit. 2019-03-27]. Dostupné z: http://storm.fsv.cvut.cz/data/files/p%C5%99edm%C4%9Bty/VK2/VK2_predn05.pdf.

[5] SOUTOR, Jiří. *Studie PPO v povodí Nihošovického potoka*. Praha, 2016. Diplomová. České vysoké učení technické v Praze Fakulta stavební. Vedoucí práce Adam Vokurka.

[6] PITKA, Roman. *NÁVRH VÍCEÚČELOVÉ NÁDRŽE V K.Ú. POPOVICE*. Brno, 2013. Diplomová. Vysoké učení technické v Brně, Fakulta stavební ústav vodního hospodářství krajiny. Vedoucí práce Doc. Dr. Ing. Petr Doležal.

[7] FENTON, John D. *Calculating flow over rectangular sharp-edged weirs: Alternative Hydraulics Paper 6* [online]. 12.5.2015, 14 [cit. 2019-03-30]. Dostupné z: http://johndfenton.com/Papers/Calculating-flow-over-rectangular-sharp-edged-weirs.pdf

[8] NODOUSHAN, Ehsan Jafari, Reza BARATI a Hossein SHAHHEYDARI. *Optimal design of labyrinth spillways using meta-heuristic algorithms: Alternative Hydraulics Paper 6. KSCE Journal of Civil Engineering* [online]. 2016, 12.5.2015, 20(1), 468-477 [cit. 2019-03-30]. DOI: 10.1007/s12205-015-0462-5. ISSN 1226-7988. Dostupné z: http://link.springer.com/10.1007/s12205-015-0462-5

[9] Jandora, J., Stara, V. a Starý, M. *Hydraulika a hydrologie. 2*. vydání. Brno: AKADEMICKÉ NAKLADATELSTVÍ CERM, s.r.o., 2011. str. 188. ISBN 978-80-7204-739-0

[10] Zachoval, Zbyněk. *Přelivy se širokou korunou pravoúhlého příčného průřezu.* Brno: VUTIUM, 2015.

[11] *JavaWorld: What is the JVM? Introducing the Java Virtual Machine* [online]. United States, 2018 [cit. 2019-04-23]. Dostupné z: https://www.javaworld.com/article/3272244/what-is-the-jvm-introducing-the-java-virtual-machine.html

[12] DRASLAR, Josef. *Ukázkový příklad vývoje Java SE aplikace dle návrhového vzoru MVC*. Praha, 2016. Bakalářská. Fakulta informatiky a statistiky Katedra informačních technologií. Vedoucí práce Doc. Ing. Alena Buchalcenová, Ph.D.

[13] Fusek, Zdeněk. *Grafické uživatelské prostředí v javafx*. Brno, 2013. Bakalářská práce. Vysoké učení technické v Brně, fakulta elektrotechniky a komunikačních technologií ústav telekomunikací.

[14] *Jetbrains.com*: idea [online]. Česká republika: JetBrains, 2019 [cit. 2019-05-01]. Dostupné z: https://www.jetbrains.com/idea/

[15] *Wikipedie: Otevřená encyklopedie*: Interpolace [online]. c2017 [cit. 2019-05-01] Dostupné z: https://cs.wikipedia.org/wiki/Line%C3%A1rn%C3%AD_interpolace

[16] ČÁPKA, David. Matlab: Lekce 1 - Seznámení s uživatelským prostředím. *www.itnetwork.cz [online]*. Praha: itnetwork.cz, ©2019, 18.6.2016 [cit. 2019-05-06]. Dostupné z: https://www.itnetwork.cz/software/matlab

[17] *KMH SOFTWARE* [online]. Brno: Vysoké učení technické v Brně, Fakulta stavební, ústav vodních staveb, 2015 [cit. 2019-05-09]. Dostupné z: http://vst.fce.vutbr.cz/cs/vysledky/software/kmh-software/

[18] *Vodohospodářství* [online]. Rybitví: vhrousa, 2015 [cit. 2019-05-09]. Dostupné z: http://www.vhrousar.cz/

[19] Program kmh 1.0 manuál: využití spolehlivostních metod při technickobezpečnostním dohledu nad vodními díly s ohledem na jejich bezpečnost v období globálních klimatických změn. In: *Vst.fce.vutbr.cz* [online]. Brno: Vysoké učení technické v Brně, 2015, září 2015 [cit. 2019-05-09]. Dostupné z: http://vst.fce.vutbr.cz/wp-content/uploads/2017/10/KMH_manual.pdf

PŘÍLOHY

PŘÍLOHA A – CD S DIPLOMOVU PRACÍ

Na cd se nachází zdrojové kódy aplikace a její spustitelný soubor. Dále je zde uveden textový soubor s požadavky na aplikaci, návodem ke spuštění a soubor s přednastavenými daty pro demonstraci aplikace.