TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta strojní

Webová aplikace pro podporu výuky předmětu Aplikovaná kybernetika

Diplomová práce

Studijní program: N2301 – Strojní inženýrství
Studijní obor: 2301T049 – Výrobní systémy a procesy

Autor práce: **Bc. Petr Karban**
Vedoucí práce: Ing. Michal Moučka, Ph.D.

Liberec 2018

Technická univerzita v Liberci
Fakulta strojní
Akademický rok: 2016/2017

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Petr Karban**
Osobní číslo:	**S16000312**
Studijní program:	**N2301 Strojní inženýrství**
Studijní obor:	**Výrobní systémy a procesy**
Název tématu:	**Webová aplikace pro podporu výuky předmětu Aplikovaná kybernetika**
Zadávající katedra:	**Katedra výrobních systémů a automatizace**

Zásady pro vypracování:

1.Seznamte se technoligií servletů a JSP.
2.Navrhněte strukturu webové aplikace s virtuálními úlohami pro podporu výuky předmětu Aplikovaná kybernetika.
3.Navrhněte virtuální úlohy simulující chování dynamických soustav v interakci se spojitými popř. nespojitými regulátory. Návrh konzultujte s vedoucím diplomové práce.
4.Virtuální úlohy naimplementujte v jazyku Java.
5.Ověřte funkčnost a správnost řešení.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **50-60 stran textu vč. příloh**

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

Seznam odborné literatury:

[1] BALÁTĚ M. Automatické řízení. Praha: BEN - technická literatura, 2003. ISBN 978-80-7300-020-2.
[2] CASTRO E.HTML, XHTML a CSS. Brno: Computer Press, 2014. ISBN 978-80-251-1531-2.
[3] HALL M. Java servlety a stránky JSP. Praha: Neocortex, 2001. ISBN 978-80-86330-06-0.
[4] HOFREITER M. Základy automatického řízení (skriptum). Praha: České vysoké učení v Praze, 2012. ISBN 978-80-7372-297-5.
[5] SHILDT H. Java 7 - Výukový kurz. Brno: Computer Press, 2013. ISBN 978-80-251-3748-2.
[6] Apache Tomcat 8 [online]. [cit. 2016-11-28]. Dostupné z: http://http://tomcat.apache.org/tomcat-8.5-doc/index.html

Vedoucí diplomové práce: **Ing. Michal Moučka, Ph.D.**
Katedra výrobních systémů a automatizace

Datum zadání diplomové práce: **1. listopadu 2016**

Termín odevzdání diplomové práce: **1. února 2018**

prof. Dr. Ing. Petr Lenfeld
děkan

L.S.

Ing. Petr Zelený, Ph.D.
vedoucí katedry

V Liberci dne 1. listopadu 2016

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

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

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

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Touto cestou bych rád poděkoval rodině a všem, kteří mne podporovali. Dále bych rád poděkoval vedoucímu diplomové práce Ing. Michalu Moučkovi Ph.D. za ochotu a rady při řešení této diplomové práce.

Abstrakt

Cílem diplomové práce je naprogramovat webovou aplikaci pro podporu výuky předmětu Aplikovaná kybernetika na katedře výrobních systémů a automatizace. Úkol byl vypracován užitím mnoha technologií. Samotné programování pak bylo realizováno ve značkovacím jazyce HTML formátovaném styly CSS. Dále pak v programovacím jazyce Java s využitím servletů a technologií pro tvorbu dynamických webových stránek JSP(Java Server Pages).

V teoretické části diplomové práce je popsána metoda Runge-Kutta 4. řádu, metoda řídícího algoritmu, řešení diferenciální rovnice N-tého řádu pomocí numerických metod, spojitá regulace, regulátory, parametry regulátorů a stabilita spojitého dynamického systému, použité technologie a přechodová charakteristika.

V praktické části je popsána struktura webové aplikace. Poté je popsán algoritmus servletu, který počítá body pro výstupní graf. Na konci je popsána webová aplikace, práce s ní a část, ve které se aplikace pro ukázku funkcionality testuje. V této části jsou ukázány i výstupy.

Klíčová slova: regulace, webová aplikace, lineární diferenciální rovnice , Runge-Kutta

Abstract

The purpose of this thesis is to program a web application to support teaching of the class of aplicated cybernetics on the department of production systems and automatization. The task was realized using many technologies. The programing itself was realized with HTML markup language modified by CSS. Also In the Java programming language and JSP(Java Server Pages) for creating a dynamic web pages.

The theoretical part of this work describes numerical method Runge-Kutta of 4th order, the method of control algorithm, solving of linear diferential equation of N-th order using numerical methods, continuous regulation, regulators, parameters of regulators, stability of continuous dynamic systems, used technologies and transitional characteristics.

In the practical part, there is described the structure of a webpage. After there is a description of servlet algorithm, which computes the points for output charts. In the end there is an introduction to web application and work with it . Also there is a part with application tests to demonstrate its functionality. In this part there are shown outputs of application.

Key words: regulation, web application, linear differential equation , Runge-Kutta

Obsah

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK 10
ÚVOD 11
1. AUTOMATICKÉ ŘÍZENÍ 13
2. SYSTÉM A JEHO POPIS 14
2.1 Lineární diferenciální rovnice 14
2.2 Přechodová funkce 15
3. REGULACE 16
3.1 Regulátor 16
3.2 Regulátory typu PID 17
3.2.1 Ideální PID regulátor 18
3.2.2 Ideální P-regulátor 19
3.2.3 Ideální I-regulátor 20
3.2.4 Derivační složka regulátoru 20
3.2.5 Regulátor PI, PD 21
4. Algoritmus řízení PID 21
4.1 Vzorkování algoritmu 22
4.2 Popis použitého algoritmu 23
5. OBYČEJNÁ DIFERENCIÁLNÍ ROVNICE N-TÉHO ŘÁDU 24
5.1 Substituce ODR řádu n 24
5.2 Výpočet soustavy ODR 25
5.2.1 Runge-Kutta 4. řádu 25
6. STABILITA SYSTÉMU 26
6.1 Nutná a postačující podmínka stability 28
7. SEŘÍZENÍ PID REGULÁTORU 29
7.1 Nastavení periody vzorkování 29
7.2 Nastavení složek P, I, D 30
7.2.1 Zieglerova-Nicholsova frekvenční metoda 30
7.2.2 Metoda pokus-omyl 31

8. TECHNOLOGIE POUŽITÉ PŘI VÝVOJI APLIKACE 32
8.1 Matlab 32
8.1.1 Simulink 32
8.2 Netbeans IDE 8.2 32
8.3 ApacheTomcat 32
8.4 Java Servlets 33
8.5 Java Server Pages(JSP) 33
8.6 HTML(HyperText Markup Language) 33
8.7 CSS(Cascading Style Sheets) 33
9. WEBOVÁ APLIKACE 35
9.1 Schéma webové aplikace 35
9.2 Popis webových stránek 36
9.2.1 Index.jsp 36
9.2.2 Ulohy.jsp 37
9.2.3 Oaplikaci.jsp 38
9.2.4 UlohaX.jsp 39
9.2.5 Vypocet.java 40
9.2.6 Výstup z aplikace 41
9.2.7 Formátování webové aplikace 41
10. ALGORITMUS PRO VÝPOČET BODŮ GRAFU 42
10.1 Vývojový diagram algoritmu 43
10.2 Rozbor metod algoritmu 46
10.2.1 Metoda vypocetkonstantaX 46
10.2.2.Metoda vypocetkonstantaX_mm0 47
10.2.3 Metoda vypocetkonstantaX_mm1 48
10.2.4 Metoda vypocetkonstantaX_mm2 48
10.2.5 Metoda vysledek_iterace 49
10.2.6 Metoda vysledek_iterace_mm0 49
10.2.7 Metoda vysledek_iterace_mm1 50

10.2.8 Metoda vysledek_iterace_mm2 51
10.2.9 Metoda regulace1 52
11. GRAFY 54
11.1 Vykreslení grafů 54
12. UKÁZKA CHODU WEBOVÉ APLIKACE 57
12.1 Volba úlohy 57
12.2 Spuštění úlohy 58
12.3 Výstup z úlohy 59
ZÁVĚR 62
SEZNAM POUŽITÉ LITERATURY 63
SEZNAM OBRÁZKŮ 65
SEZNAM TABULEK 66
PŘÍLOHA 67

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

$a_0,\ a_1,\ a_n,\ b_0,\ b_1,\ b_m$	- koeficienty lineární diferenciální rovnice
$G(s)$	- obrazový přenos systému
T	- krok
$k_i,\ k_1,\ k_2,\ k_3,\ k_4$	- konstanty Runge-Kutta 4.řádu
$u(t)$	- akční veličina(vstup do systému)
$U(s)$	- Laplaceův obraz vstupní veličiny
$y(t)$	- regulovaná veličina(výstup ze systému)
y_n, y_i	- výsledek *n*-tého kroku, výsledek *i*-tého kroku
y_{n+1}	- výsledek *n + 1*-tého kroku
$Y(s)$	- Laplaceův obraz výstupní veličiny
$\eta(t)$	- jednotkový skok
$w(t)$	- žádaná hodnota
$e(t)$	- regulační odchylka
K_R	- statické zesílení regulátoru
LDR	- lineární diferenciální rovnice
ODR	- obyčejná diferenciální rovnice
RK	- Runge–Kutta
T_I	- integrační časová konstanta
r_I	- integrační konstanta
T_D	- derivační časová konstanta
r_D	- derivační konstanta

ÚVOD

Cílem diplomové práce je vytvořit webovou aplikaci pro podporu výuky předmětu aplikovaná kybernetika na katedře výrobních systémů a automatizace. Algoritmus aplikace pracuje s dynamickým systémem a regulátorem. Jejich spojením vzniká regulovaná soustava. Jako vstupní dynamické systémy jsou dvě virtuální úlohy: řízení teploty na žehličce a řízení zkrácení pneumatického svalu. Ve webové aplikaci si jednu z úloh zvolíme. Každá úloha má pevně dané a editovatelné parametry. Editovatelné parametry navolíme. Po spuštění úlohy se numerickou metodou spočítají body grafu popisující regulovanou soustavu. Tyto body se nakonec vykreslí v grafu.

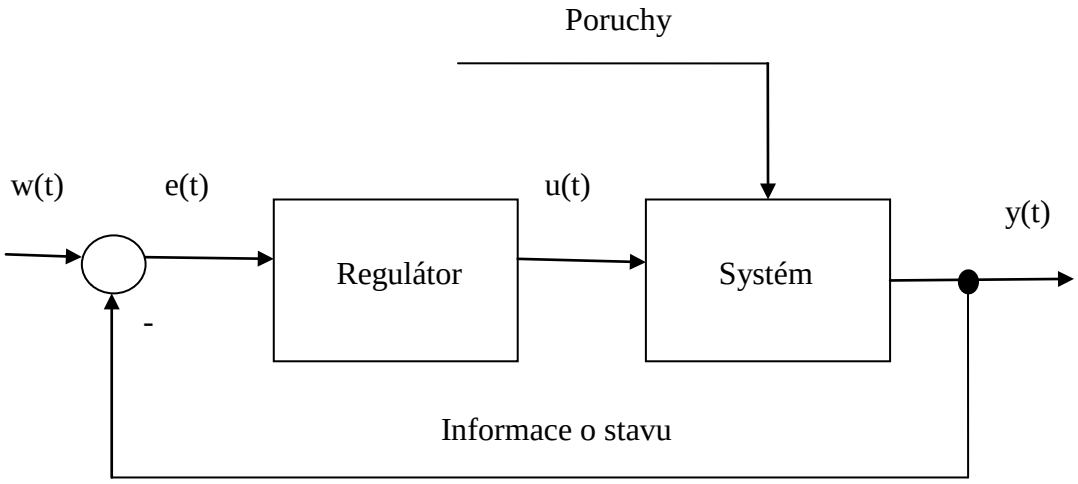
Text je rozdělen do dvou částí. Na teoretickou část a na praktickou část. V teoretické části je popsán princip regulace , dynamický systém a regulátor. Dále je popsáno teoretické pozadí algoritmu. To znamená teorie na základě, které byly tvořeny algoritmy. V závěru teoretické části je pak popsána stabilita , seřízení regulátoru a použité technologie.

V praktické části je popsána webová aplikace a to jakým způsobem jsou jednotlivé webové stránky provázány. Je tam také uveden popis webových stránek popis popis. V další části pak algoritmus a zdrojový kód algoritmu pro výpočet bodů grafu. Dále pak zdrojový kód pro vykreslení grafů. V závěru praktické části je pak ukázán chod webové aplikace spolu s výstupy pro ověření funkčnosti aplikace.

Teoretická část

1. AUTOMATICKÉ ŘÍZENÍ

Při řešení úloh z automatického řízení se setkáváme s tím, že řešíme dynamické vlastnosti regulačního obvodu. Regulační obvod dostaneme spojením regulátoru a regulované soustavy. V regulačním obvodu dochází k regulaci soustavy. Regulace je v podstatě udržování technologické veličiny na požadované hodnotě. Tuto funkci realizuje regulátor. Regulátor může být spojitý anebo nespojitý. Dále v této diplomové práci budeme uvažovat regulaci spojitou. U spojitého regulátoru existuje nepřetržitá trvalá vazba v čase mezi veličinami řízeného obvodu. Princip regulace je znázorněn na obr. 1.



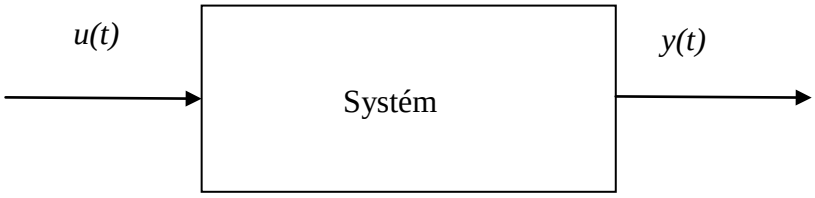
Obr. 1.: Princip regulace

V naší úloze uvažujeme ideální regulátory a systém bez vstupujících poruch. Do regulované soustavy zadáme požadovanou hodnotu výstupu $w(t)$. Tuto hodnotu žádáme na výstupu $y(t)$.

Toho aby se $w(t) \approx y(t)$ dosáhneme tím, že do systému vstupuje jako vstupní veličina $u(t)$. To je výstupní veličina z regulátoru. Regulovaná soustava díky zpětné vazbě dostává informaci o aktuální hodnotě regulované veličiny y (t) a z rovnice (1) dopočítá vstup do regulátoru regulační odchylku e(t) jako v rovnici (2) [1].

2. SYSTÉM A JEHO POPIS

Systém si lze představit jako blok, jako na obr.2., v němž jsou soustředěny jeho dynamické vlastnosti. Systém může být popsán vnějším nebo vnitřním popisem.



Obr. 2.: Blokové schéma systému

Dále budeme uvažovat popis vnější. Ten vyjadřuje dynamické vlastnosti reakcí mezi vstupem $u(t)$ a výstupem $y(t)$soustavy a může být popsán lineární diferenciální rovnicí(LDR), přechodovou charakteristikou atd.[2].

2.1 Lineární diferenciální rovnice

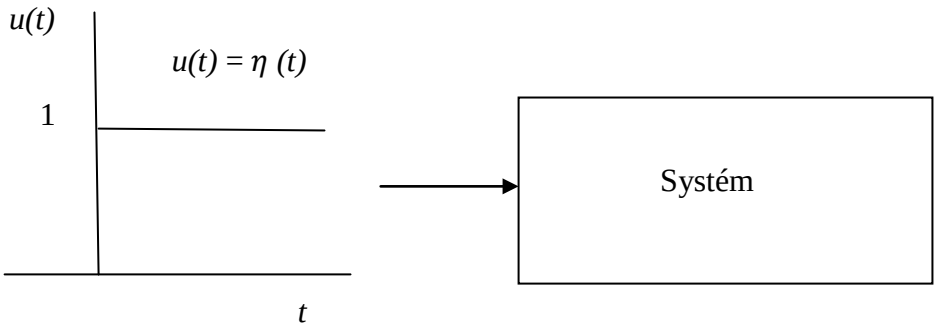
Lineární spojitý system nebo regulační člen se vstupem $u(t)$ a výstupem $y(t)$ je obecně popsán LDR:

$$a_n \cdot y^{(n)}(t) + \ldots + a_1 \cdot y'(t) + a_0 \cdot y(t) = b_m \cdot u^{(m)}(t) + \ldots + b_1 \cdot u'(t) + b_0 \cdot u(t) \qquad (1)$$

Musí být vždy splněna podmínka fyzikální realizovatelnosti $m \leq n$. Diferenciální rovnici systému získáváme obvykle tak, že uvedeme fyzikální vztahy a zákony v systému a vyeliminujeme všechy veličiny mimo vstupní a výstupní [2]. Webová aplikace využívá LDR pro popis systému.

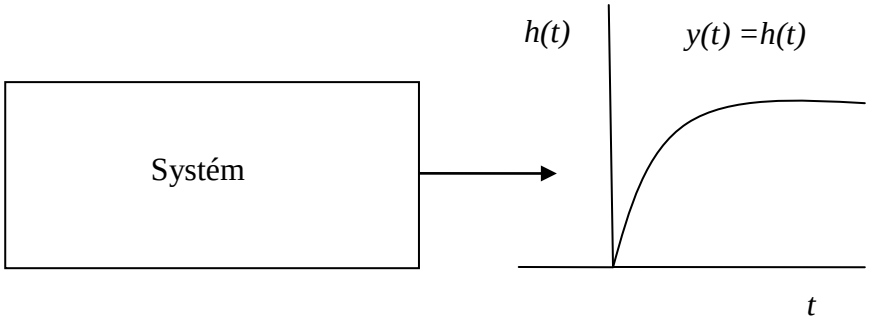
2.2 Přechodová funkce

Přechodová funkce popisuje odezvu systému na jednotkový skok $\eta(t)$ na vstupu při nulových počátečních podmínkách jako na obr. 3.[2]



Obr. 3.: Jednotkový skok vstupující do systému

Značíme ji $h(t)$. Její graf je přechodová charakteristika jako na obr.4.[2]



Obr. 4.: Přechodová charakteristika

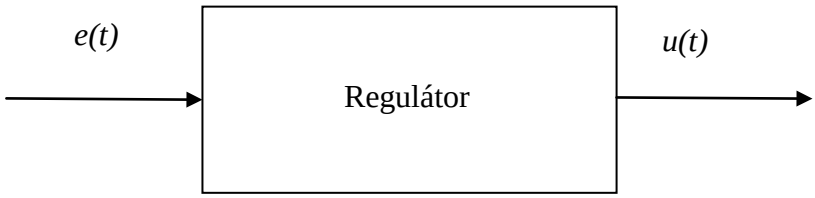
Webová aplikace je schopna tuto charakteristiku v testovacím režimu, během testování, pro dané úlohy vykreslit.

3. REGULACE

Regulace je proces při, kterém se za pomocí úpravy vstupu $u(t)$ do systému upraví výstup ze systému $y(t)$, tak aby byl shodný s požadovanou hodnotou $w(t)$. Je provedena pomocí regulátoru jako na obr. 1.

3.1 Regulátor

Regulátor je možné znázornit pomocí blokového schématu jako na obr. 5.



Obr. 5.: Blokové schéma regulátoru

Kde $e(t)$ je vstupující regulační odchylka jako v rovnici (2) a $u(t)$ je výstupní veličina, která dále vstupuje do regulovaného systému jako v kapitole 2.[1]

$$e(t) = w(t) - \mathrm{y}(t) \tag{2}$$

3.2 Regulátory typu PID

Ve webové aplikaci jsou použity regulátory typu PID. Jsou to regulátory nepřímé(potřebují k činnosti pomocnou energii), spojité(vstupní i výstupní signál spojitou funkcí času, tj. může se měnit v každém časovém okamžiku) a lineární. [3]

V závislosti, které složky jsou použity při regulaci, rozlišujeme následujících pět typů regulátorů

- P-regulátor(proporcionální regulátor)
- I-regulátor(integrační regulátor)
- PI-regulátor(proporcionálně-integrační regulátor)
- PD- regulátor(proporcionálně-derivační regulátor)
- PID-regulátor(proporcionálně-integračně-derivační regulátor)

3.2.1 Ideální PID regulátor

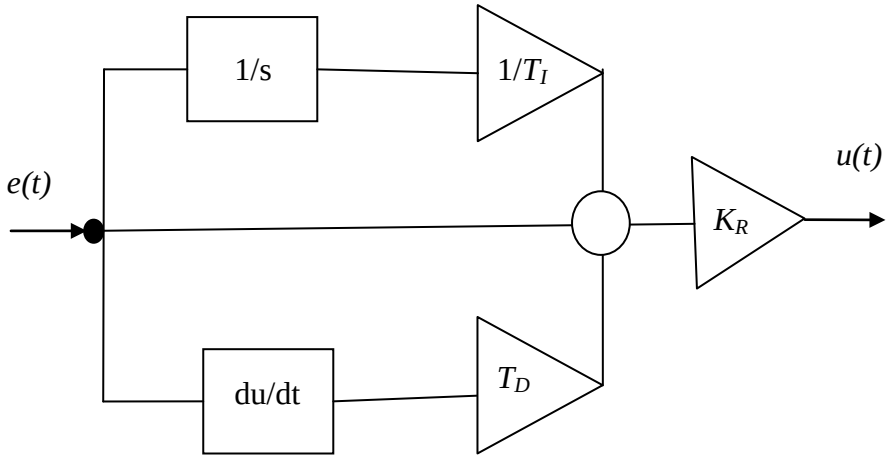
Regulátor s proporcionální(P), integrační(I) a derivační(D) složkou. Lze popsat vztahem:

$$u(t) = K_R \cdot \left(e(t) + \frac{1}{T_I} \cdot \int_0^t e(t)dt + T_D \cdot e'(t)\right) + u(0)\,, \qquad (3)$$

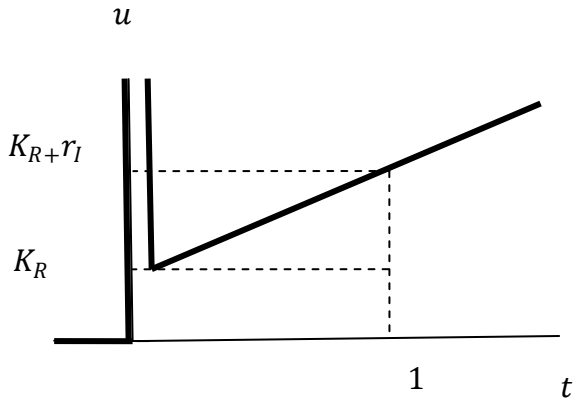
kde integrační časová konstanta $T_I = \frac{K_R}{r_I}$ a derivační časová konstanta $T_D = \frac{r_D}{K_R}$. [3]

Volba složek regulátoru je popsána v kapitole 7.

Na obr. 6 Je znázorněno blokové schéma PID regulátoru a na obr .7. je znázorněna přechodová charakteristika PID regulátoru.



Obr. 6.: Blokové schéma PID regulátoru



Obr.7.: Přechodová charakteristika PID regulátoru

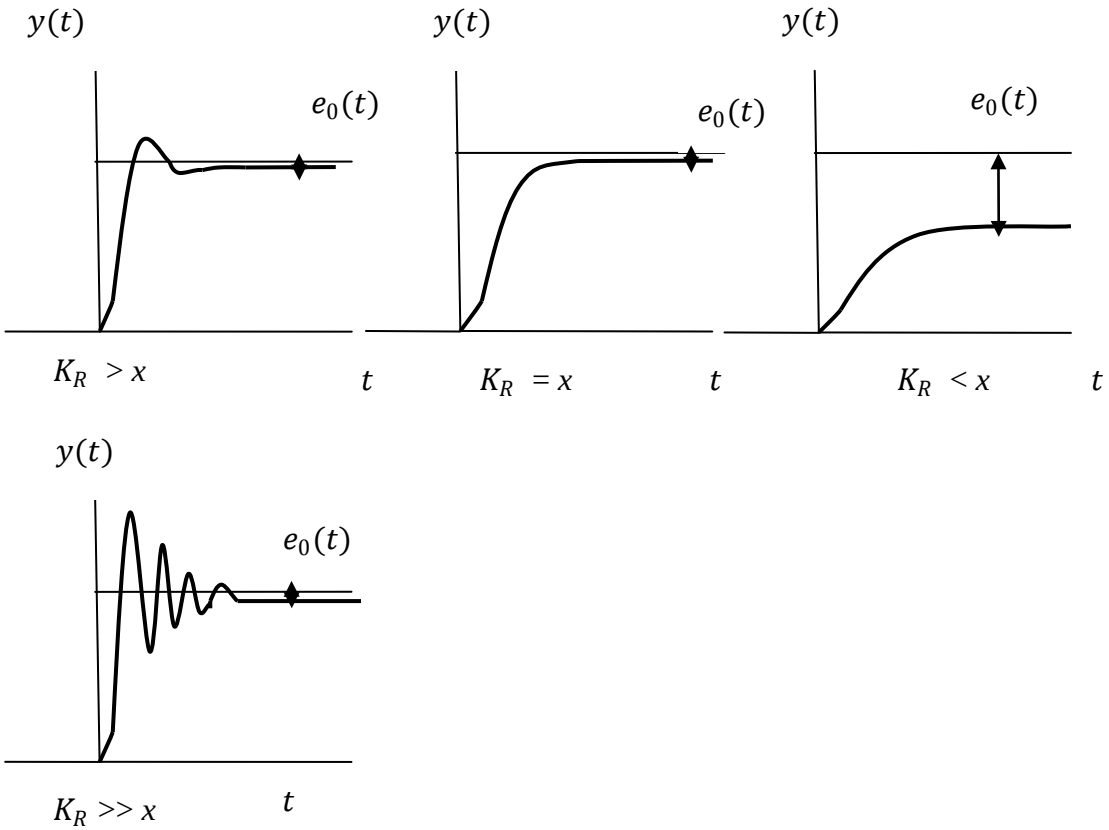
3.2.2 Ideální P-regulátor

Ideální P-regulátor nastavuje hodnotu akční veličiny $u(t)$ úměrně k regulační odchylce $e(t)$. Jeho chování lze popsat vztahem:

$$u(t) = K_R \cdot e(t) \tag{4}$$

Samostatný P-regulátor nedokáže regulovat při řízení statického systému na přesnou žádanou hodnotu. Má tzv. trvalou regulační odchylku $e_0(t)$ [3] pro $K_R = x$, , kde x je optimální hodnota, při které dosáhneme během regulace statického systému nejoptimálnějšího možného stavu a K_R zesílení P- regulátoru.

Při vysokých hodnotách $K_R > x$ je funkce „tlačena“ k požadované hodnotě silněji se snahou dosáhnout požadované hodnoty dříve. Při nízkých hodnotách $K_R < x$ naopak je přechod k požadované hodnotě pomalejší. Při vysokém zesílení K_R však může dojít k překmitu(tzn. překročení požadované hodnoty), což nemusí být u řady aplikací žádoucí. Při velmi vysokém zesílení $K_R \gg x$ může dojít až k rozkmitání soustavy a ztrátě stability(stabilita kap.6).



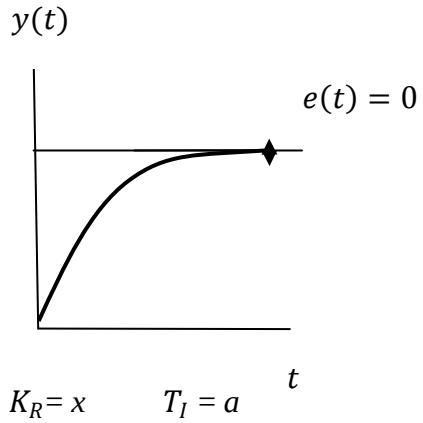
Obr. 8.: Příklady průběhů pro P-regulátor-statická soustava

3.2.3 Ideální I-regulátor

Ideální I-regulátor nastavuje hodnotu akční veličiny úměrně integrálu regulační odchylky $e(t)$. Jeho chování lze popsat vztahem:

$$u(t) = \frac{1}{T_I} \cdot \int_0^t e(t)dt \; + u(0) \tag{5}$$

Oproti P-regulátoru ideální I-regulátor zasahuje pozvolna, avšak je schopen dosáhnout nulové regulační odchylky $e(t)$. [3] $T_I = \; a$ je vhodně zvolená složka T_I. Volba složky T_I bude dále popsána v kapitole 7.



Obr. 9.: Dosažení nulové odchylky

3.2.4 Derivační složka regulátoru

Slouží k předpovědi budoucího vývoje regulační odchylky. Upravuje akční zásah $u(t)$. Má stabilizující vliv na regulační pochod. Při rychlých změnách regulační odchylky by derivační složka způsobovala velké a nežádoucí změny v $u(t)$. Derivační složka se upravuje pomocí filtru.[3]

3.2.5 Regulátor PI, PD

Regulátor PI a PD vzniká odebráním derivační resp. integrační složky z regulátoru PID

Vztah pro popis PI- regulátoru:

$$u(t) = K_R \cdot \left(e(t) + \frac{1}{T_I} \cdot \int_0^t e(t)dt\ \right) + u(0) \quad (6)$$

Vztah pro popis PD-regulátoru:

$$u(t) = K_R \cdot \left(e(t) + T_D \cdot e^{'}(t)\right) \quad (7)$$

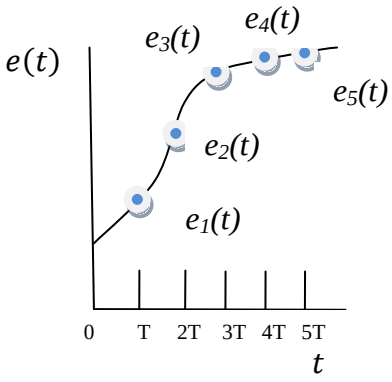
[3].

4. Algoritmus řízení PID

Algoritmus řízení PID na základě výpočtu s využitím zpětné vazby a odchylky $e(t)$ dopočítává akční zásah $u(t)$, který následně vstupuje do regulovaného systému. (kap. 3, kap. 2.). Algoritmus má volitelné parametry K_R, T_I, T_D (kap. 3)

4.1 Vzorkování algoritmu

Číslicový regulátor vyhodnocuje odchylku $e(t)$ diskrétně. V $t = k \cdot T$, kde $k = 0,1,2$ ... a T je perioda vzorkování.



Obr. 10.: Vzorkování

Hodnoty $e_1(t), e_2(t), e_3(t), e_4(t), e_5(t)$ popisují odchylky v časech $t = k \cdot T$ pro $k = 1,2,3,4,5$.

Volba vzorkovací periody by měla být taková, aby rozdíl v kvalitě regulace při použití podobného spojitého regulátoru nebyl větší než 15%. [1] Hodnotu T je možno ve webové aplikaci nastavit. Nastavit lze také $t_{max} = k_{max} \cdot T$.

4.2 Popis použitého algoritmu

Algoritmus řízení použitý ve webové aplikaci má tvar(algoritmus regulátoru v přírustkovém tvaru):

$$\Delta u(k \cdot T) = K_R \cdot (-y(k \cdot T) + y[(k-1) \cdot T] + \frac{T}{T_I} \cdot e(k \cdot T) + + \frac{T_D}{T} \cdot (-y(k \cdot T)$$

$$+2 \cdot y[(k-1) \cdot T] - y \cdot [(k-2) \cdot T])), \quad (8)$$

jako v [4].

Pro výpočet akčního zásahu $u(t)$, je třeba použít vztah:

$$u(t) = \Delta u(k \cdot T) + u[(k-1) \cdot T] \quad (9)$$

Volitelné parametry algoritmu jsou K_R, T_D, T_I a algoritmus využívá informace o výstupní veličině $y(t)$: $y(k \cdot T)$, $y[(k-1) \cdot T]$, $y \cdot [(k-2) \cdot T]$ a o odchylce $e(k \cdot T)$.

5. OBYČEJNÁ DIFERENCIÁLNÍ ROVNICE N-TÉHO ŘÁDU

Obyčejné diferenciální rovnice jsou matematické rovnice, které obsahují neznámou funkci jedné nezávislé proměnné a její derivace. Třídou obyčejných diferenciálních rovnic jsou LDR jako v kap. 2.1.

5.1 Substituce ODR řádu n

Obyčejné diferenciální rovnice vyššího řádu je možné řešit rozkladem na soustavu rovnic řádu prvního jako v [5]. Mějme tvar rovnice:

$$y^{(n)} = f\left(x, y, y', \dots, y^{(n-1)}\right), \tag{10}$$

S počátečními podmínkami pro rovnici (10):

$$y(x_0) = y_0, y'(x_0) = y_0'(x_0) \text{ až } y^{(n-1)}(x_0) = y_0^{(n-1)}. \tag{11}$$

Označíme-li $y_1 = y$, $y_2 = y'$, $y_n = y^{(n-1)}$, bude platit také $y_1' = y_2, y_2' = y_3, \dots$

$y_{(n-1)}' = y_n$. Tuto substituci lze použít k převodu na tvar:

$$y_n' = f(x, y_1, y_2, \dots, y_n). \tag{12}$$

Rovnice (10) představuje soustavu n diferenciálních rovnic prvního řádu:

$$\begin{aligned} y_1' &= y_2, & y_1(x_0) &= y_0, \\ y_2' &= y_3, & y_2(x_0) &= y_0', \\ &\vdots & &\vdots \\ y_n' &= f_n(x, y_1, y_2, \dots, y_n), & y_n(x_0) &= y_0^{(n-1)} \end{aligned} \tag{13}$$

Substitucí jsme tedy získaly z ODR *n*-tého řádu soustavu ODR *n*-rovnic řádu prvního.

5.2 Výpočet soustavy ODR

Metody pro výpočet soustavy ODR jsou buď vícekrokové nebo jednokrokové.

V algoritmu byla použita metoda jednokroková. Ta využívá k výpočtu výstupní hodnoty systému $y(t)$, $t = k \cdot T$ pouze předchozí krok $t = (k-1) \cdot T$.

Konkrétně pak byla zvolena metoda Runge-Kutta 4.řádu.

5.2.1 Runge-Kutta 4. řádu

V rovnici (13) byla popsána soustava diferenciálních rovnic prvního řádu po rozkladu z řádu n-tého. Tím nám vznikla soustava ODR prvního řádu.

Soustava ODR prvního řádu s počátečními podmínkami bude jako v [5] rozepsána:

$$\begin{aligned} y_1' &= f_1(x, y_1, y_2, \dots, y_n), & y_1(x_0) &= \eta_1, \\ y_2' &= f_2(x, y_1, y_2, \dots, y_n), & y_2(x_0) &= \eta_2, \\ &\vdots & &\vdots \\ y_n' &= f_n(x, y_1, y_2, \dots, y_n), & y_n(x_0) &= \eta_n. \end{aligned} \tag{14}$$

Vektorově může být popsána jako v [5]:

$$\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y}), \quad \boldsymbol{y}(x_0) = \boldsymbol{\eta}. \tag{15}$$

, kde $\boldsymbol{y} = (y_1, \dots, y_n)^T$, $\boldsymbol{f} = (f_1, \dots, f_n)^T$ a $\boldsymbol{\eta} = (\eta_1, \dots, \eta_n)^T$.

Počáteční podmínky ve webové aplikaci jsou neměnně nastaveny na

$\eta_1 = 0, \dots, \eta_n = 0.$

Vektorový tvar Rungovy-Kuttovy metody 4. řádu pro soustavu je:

$$\boldsymbol{y}_{i+1} = y_i + \frac{1}{6} \cdot T(\boldsymbol{k_1} + 2 \cdot \boldsymbol{k_2} + 2 \cdot \boldsymbol{k_3} + \boldsymbol{k_4}), \tag{16}$$

$$\mathbf{k_1} = \boldsymbol{f}(x_i, \boldsymbol{y_i})$$

$$\mathbf{k_2} = \boldsymbol{f}(x_i + \frac{1}{2} \cdot h, \boldsymbol{y_i} + \frac{1}{2} \cdot h \cdot \mathbf{k_1})$$

$$\mathbf{k_3} = \boldsymbol{f}(x_i + \frac{1}{2} \cdot h, \boldsymbol{y_i} + \frac{1}{2} \cdot h \cdot \mathbf{k_2})$$

$$\mathbf{k_4} = \boldsymbol{f}(x_i + h, \boldsymbol{y_i} + h \cdot \mathbf{k_3})$$

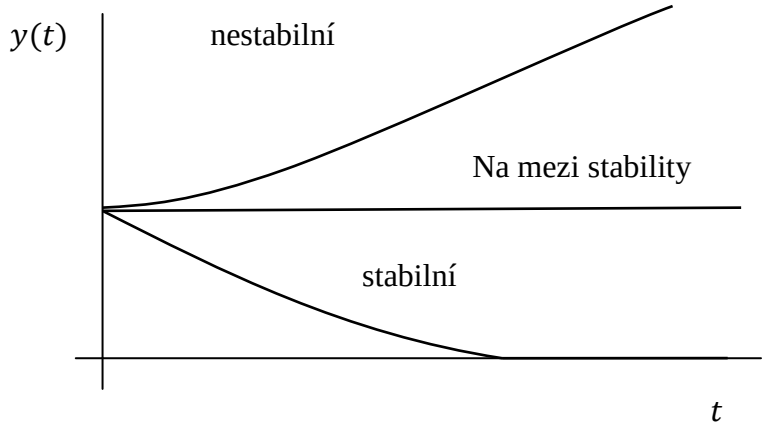
6. STABILITA SYSTÉMU

Stabilita je schopnost systému se po odeznění působících vstupů $u(t)$ vrátit do rovnovážného stavu.Matematicky lze popsat:

$$\lim_{t\to\infty} y\,(t) = 0$$

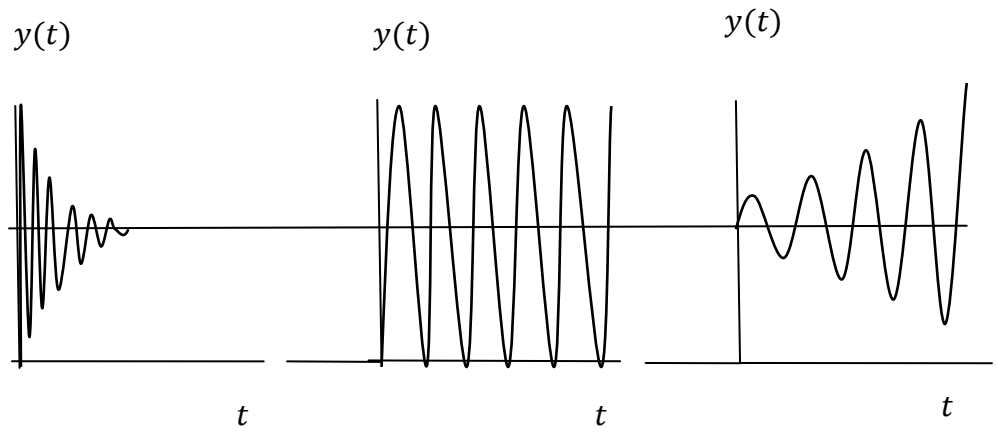
Dle chování systému po odeznění $u(t)$ (např. využitím přechodové funkce) lze rozlišit systémy popřípadě regulační obvody na stabilní, na mezi stability a nestabilní. Jejich chování je popsáno na obr. 11

Systémy na mezi stability se považují za stabilní. [6][1]



Obr. 11.: Lineární nekmitavý průběh 1. řádu - stabilita

Níže jsou příklady průběhů kmitavých systémů s trvale působícím vstupem $u(t)$



Stabilní Na mezi stability Nestabilní

Obr. 12.: Kmitavé průběhy - stabilita

6.1 Nutná a postačující podmínka stability

Charakteristická rovnice pro rovnici (1) je po zanedbání pravé strany:

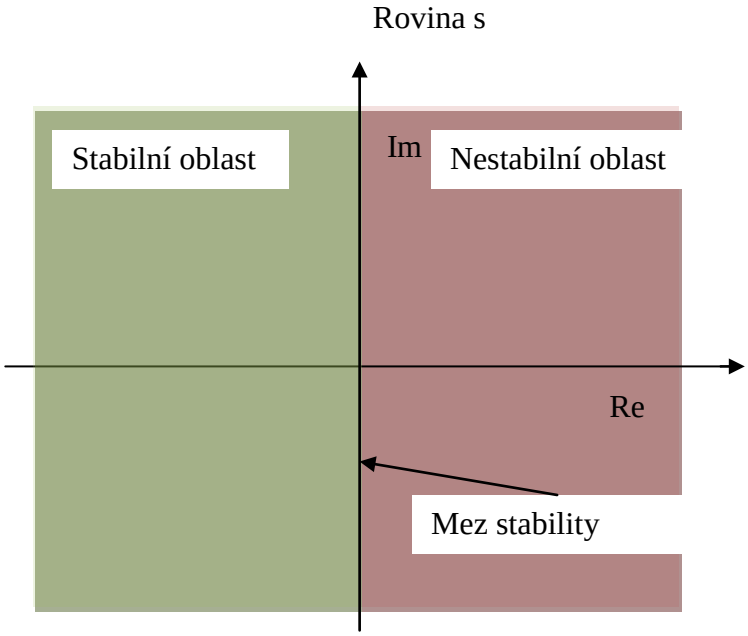
$$a_n \cdot s^n + \ ... + a_1 \cdot s + \ a_0 \ = \ 0 \qquad (17)$$

$$Re\ s_i \leq 0, i = 1, \dots, n$$

Rozložíme kořeny charakteristické rovnice v komplexní rovině kořenů jako na obr.13.

Jestliže, lze kořeny charakteristické roviny vyčíslit, tak použijeme nutnou a postačující podmínku stability.

Regulační obvod je stabilní právě tehdy, když všechny kořeny z charakteristické rovnice mají záporné reálné části, tedy leží-li v levé komplexní polorovině.



Obr. 13.: Znázornění oblasti stability

V případě , že nelze kořeny vyčíslit , tak je třeba použít kritéria stability. To jsou pravidla umožňující rozhodnout o stabilitě bez přímého výpočtu kořenů charakteristické rovnice. [7]

7. SEŘÍZENÍ PID REGULÁTORU

Ve webové aplikaci je možné nastavit periodu vzorkování T , zesílení proporcionální složky K_R, zesílení integrační složky T_I a zesílení derivační složky T_D.

7.1 Nastavení periody vzorkování

Perioda vzorkování T při velmi nízké hodnotě zvyšuje výpočetní náročnost .

Pro volbu periody vzorkování existuje řada vztahů, například:

$$T = \left(\frac{1}{15}\ a\check{z}\ \frac{1}{6}\right) \cdot t_{95}, \tag{18}$$

Kde t_{95} je čas potřebný k dosažení 95% ustálené hodnoty přechodové charakteristiky nekmitavého regulačního obvodu(bez dopravního zpoždění).[3]

Více o nastavení periody vzorkování je popsáno v [1].

7.2 Nastavení složek P, I, D

V této kapitole je probráno několik základních postupů pro nastavení PID regulátorů. Zieglerova – Nicholsonova frekvenční metoda a metoda pokus – omyl spolu s příkladem jejího zefektivnění.

7.2.1 Zieglerova-Nicholsova frekvenční metoda

Při $T_I = 0$ a $T_D = 0$ je postupně zvětšováno K_R až do chvíle než se obvod dostane na mez stability a objeví se netlumené kmity s konstantní amplitudou. Frekvence těchto kmitů je rovna kritické frekvenci. Z naměřeného záznamu regulované veličiny určíme periodu ustálených kmitů T_{180} a příslušné kritické zesílení P regulátoru K_K :

$$K_K = K_{180} \tag{19}$$

K_K je hodnotou zesílení regulované soustavy na frekvenci:

$$f_{180} = \frac{1}{T_{180}}\,. \tag{20}$$

Parametry K_R, T_I ,T_D určíme z tab. 1. [8].

Tab.1.: Pravidla pro určení parametrů

Regulátor	K_R	T_I	T_D
P	$0{,}5 \cdot K_K$		
PI	$0{,}45 \cdot K_K$	$0{,}85 \cdot T_{180}$	
PID	$0{,}6 \cdot K_K$	$0{,}5 \cdot T_{180}$	$0{,}125 \cdot T_{180}$

7.2.2 Metoda pokus-omyl

Hodnoty parametrů jsou voleny podle tvaru odezvy na vstup $u(t)$ o požadované hodnotě. Dle odezvy se subjektivně posoudí zda-li jsou parametry vhodné.

Pro zefektivnění této metody existuje řada pravidel. Například je možné postupovat podle následujících kroků jako v [8]:

1. Vypněte integrační a derivační složku(nastavte $T_I \to \infty$ a $T_D = 0$). Postupně zvyšujte K_R až vzniknou trvalé kmity. Poté zmenšete K_R na polovinu.
2. Pomalu zmenšujte T_I až do chvíle než vzniknou trvalé kmity. Poté T_I zvětšete třikrát.
3. Pomalu zvětšujte T_D až do chvíle než nastanou trvalé kmity. Pak T_D zmenšete třikrát.

8. TECHNOLOGIE POUŽITÉ PŘI VÝVOJI APLIKACE

V této kapitole jsou popsány technologie, které byly použity pro tvorbu webové aplikace.

8.1 Matlab

Matlab je vysoce výkonný jazyk pro technické výpočty. Integruje výpočty, vizualizaci a programování do jednoduše použitelného prostředí. Patří mezi univerzální matematické balíčky. Základním datovým typem je dvourozměrné pole. Slouží pro řešení technických problémů, speciálně takových, které vedou na vektorovou či maticovou formulaci.

Jako nadstavba Matlabu existuje Simulink. Matlab byl použit pro porovnání výstupů při výpočtu diferenciální rovnice metodou Runge-Kutta s výstupy z webové aplikace.[9]

8.1.1 Simulink

Simulink je určen na řešení(simulaci) chování dynamického systému. Je třeba znát matematický popis systému. Lze s jeho pomocí řešit časové průběhy výstupních veličin(i vnitřních) v závislosti na časovém průběhu veličin vstupních a počátečním stavu. Způsob zápisu modelu je grafický. Z nabídky příslušné knihovny se myší přetahují bloky následně se na tyto bloky připojí vstupy a výstupy. Výsledek simulace(časový průběh řešení) se zobrazuje nejčastěji graficky pomocí standardních bloků. Například jako zobrazení typu osciloskop či XY graf. Simulink byl použit k tvorbě regulačních obvodů a porovnání jejich výstupu s výstupem webové aplikace.[9]

8.2 Netbeans IDE 8.2

Vývojové prostředí Netbeans IDE slouží k vývoji desktopových, mobilních a webových Java aplikací. Slouží k vývoji aplikací v HTML5, JavaScript a CSS. Nabízí také velkou paletu nástrojů pro vývoj pro vývojáře v PHP a C/C++. Je zdarma a open source. Použil jsem ho pro psaní kódu v Java, HTML a CSS. [10]

8.3 ApacheTomcat

Apache Tomcat je webový server. Je vyvíjený jako open source a je založený na jazyce Java, Java servlets, Java Server Pages a Enterprise Java Beans. Byl použit k testování funkčnosti webové aplikace. [11]

8.4 Java Servlets

Java Servlets jsou na platformě nezávislé moduly HTTP na straně severu. Z technického hlediska je servletem každá java třída , která implementuje rozhraní javax.servlet. Servlet slouží k obsloužení HTTP požadavku a vygenerování libovolné odpovědi. V první fázi web server nahraje třídu servletu. Servlet se následně inicializuje. Po inicializaci může sevlet obsluhovat klientské požadavky. Každý požadavek je v odděleném vlákně.Pro generování struktury HTML stránek a jejich prezentaci však lépe slouží JSP. [14]

8.5 Java Server Pages(JSP)

JSP stránka je textový dokument obsahující statická data(HTML,SVG,WML a XML) a element tvořící dynamický obsah. V první fázi je JSP stránka přeložena na servlet a zkompilována. Ve druhé fázi již obsluhuje požadavky stejně jako servlet. JSP má stejnou strukturu jako HTML stránka. Je pouze obohaceno o JSP element.[14]

8.6 HTML(HyperText Markup Language)

Jazyk pro psaní webových stránek. Webové stránky jsou propojeny pomocí hypertextu. HTML ve zdrojovém kódu obalí text značkami, které specifikují webovému prohlížeči jak tento text zpracovat.[12]

8.7 CSS(Cascading Style Sheets)

Tzv. Kaskádové styly(Cascading style sheets) slouží k definování formy a vzhledu webové stránky. Oddělují strukturu formy od HTML a na HTML ponechávají strukturu obsahu. [13]

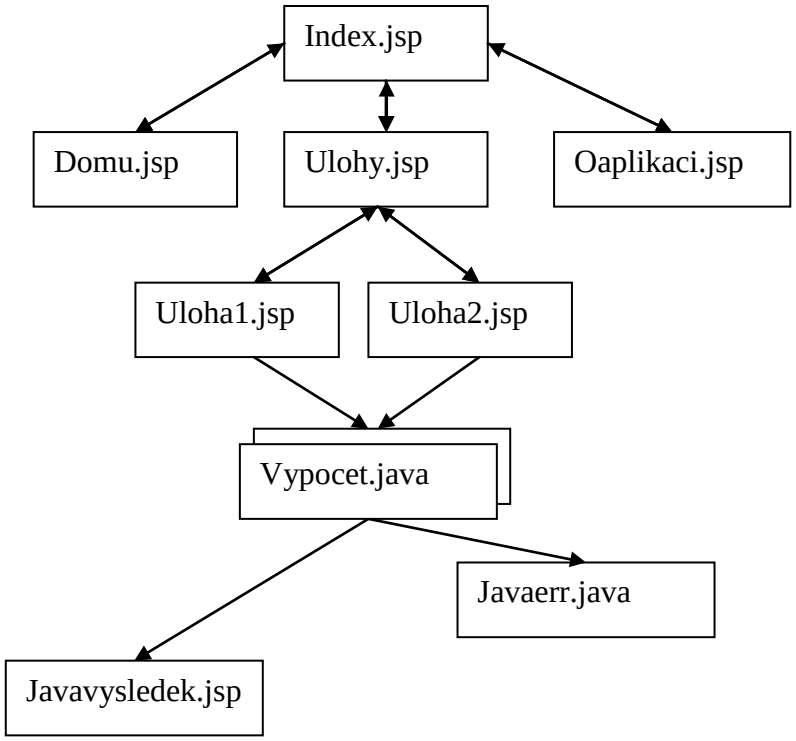
Realizační část

9. WEBOVÁ APLIKACE

V následujícím textu je popsána webová aplikace pro simulaci chování dynamických soustav v interakci se spojitými regulátory. Technologie použité při vývoji aplikace jsou JSP, Servlet, HTML, CSS. Tyto technologie byly již popsány v kap. 8.

9.1 Schéma webové aplikace

Ve schématu na obr. 14 je vyobrazeno schéma webu popisující vzájemnou vazbu jednotlivých JSP stránek a Servletu. V dalším textu jsou pak jednotlivé stránky rozebrány a popsán jejich účel. Dále v textu pak bude popsán samotný algoritmus pro výpočet výstupní hodnoty $y(t)$ v reakci na vypočtenou vstupní hodnotu $u(t)$.



Obr. 14.: Schéma webové aplikace

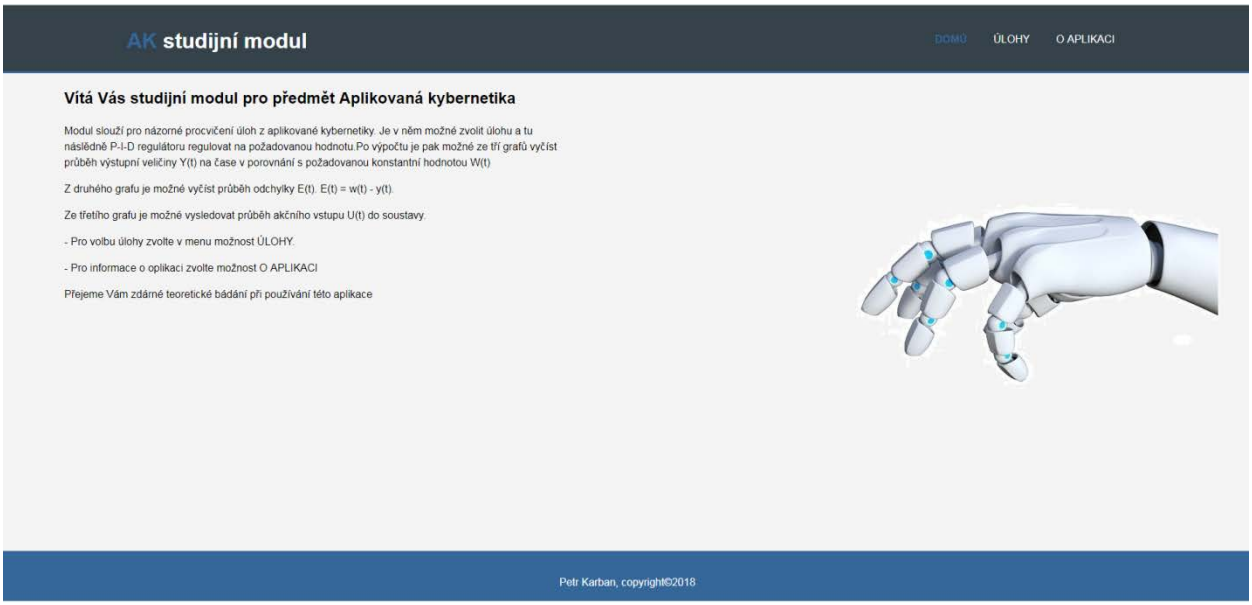
9.2 Popis webových stránek

Popis je členěn tak, aby odpovídal postupu od vrcholu schématu z obr. 14 dolů, tzn. index.jsp je popsán jako první. V každé úrovni níže pak zleva doprava tedy pro druhý řádek zleva Domu.jsp, Ulohy.jsp, Oaplikaci.jsp atp.

9.2.1 Index.jsp

Index. jsp je psán technologií HTML a JSP s použitím CSS pro formátování struktury webové stránky. Toto platí pro všechny stránky s koncovkou .jsp uvedené dále. V případě výjimky bude seznam technologií opět uveden a upřesněn.

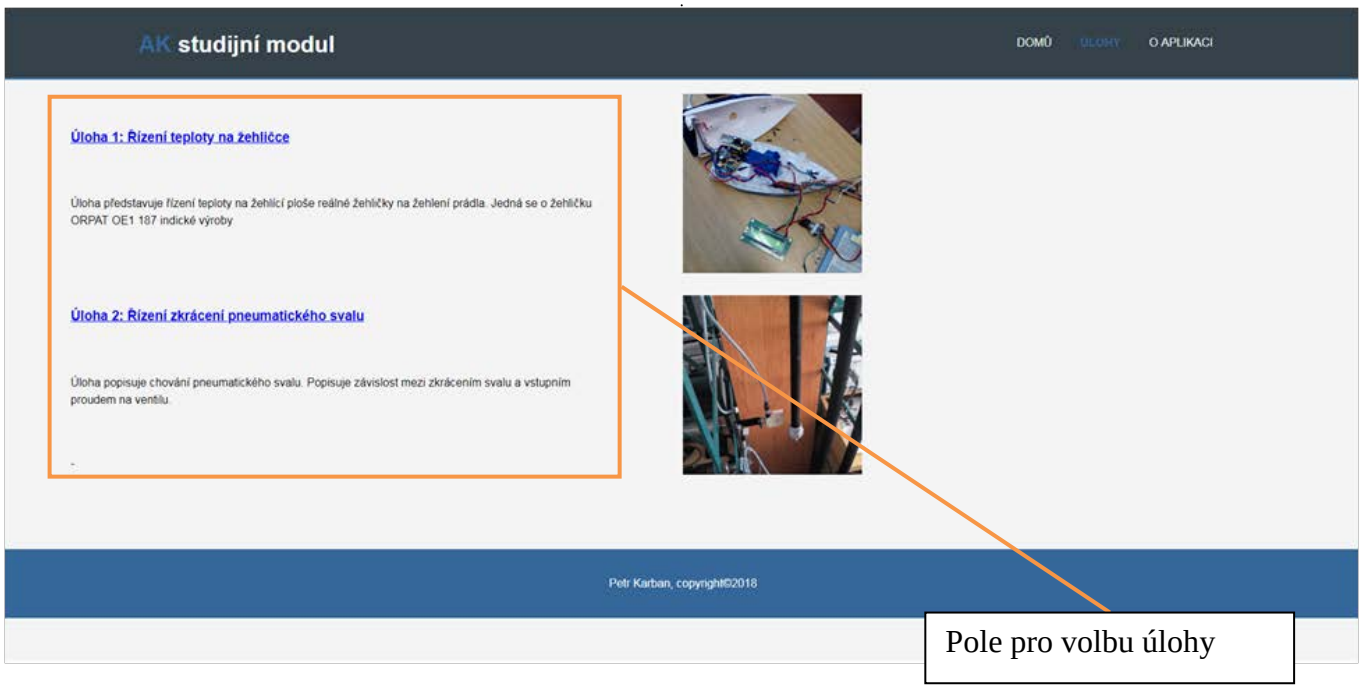
Stránka index.jsp se načte při spuštění webové aplikace jako první. Použité tagy značkovacího jazyka stejně tak jako jejich význam a použití je možné najít ve zdroji[12]. V tomto zdroji je možné najít také informace o postupech formátování stránek pomocí CSS. Případně lze i ověřit, zda-li je použitý tag přístupný pro všechny typy prohlížečů a je-li podporován nejnovější technologií.



Obr. 15.: Index.jsp

9.2.2 Ulohy.jsp

Ulohy.jsp slouží jako přehled jednotlivých úloh. Představuje tak možnosti, které může uživatel aplikace zvolit volbou přes hypertextové odkazy. Uživatel má možnost zvolit si mezi dvěma úlohami: řízení teploty na žehličce a řízení zkrácení pneumatického svalu.



Obr. 16.: Ulohy.jsp

Pole pro volbu úlohy je realizováno následujícím kódem v HTML:

```
<a  href="uloha1.jsp " >  <h3>Úloha 1: Řízení teploty na žehličce</h3></a>

<a  href="uloha2.jsp " ><h3>Úloha 2: Řízení zkrácení pneumatického svalu</h3></a>
```

9.2.3 Oaplikaci.jsp

Na této webové stránce je popsána motivace pro vznik aplikace společně s kontaktem pro návrhy a nápady pro další zlepšení a rozvoj aplikace.

9.2.4 UlohaX.jsp

Na této webové stránce se za označení X se dosazuje číslo úlohy(např. pro 1 se soubor nazývá uloha1.jsp). Na této stránce se vyobrazí bližší popis úlohy spolu s parametry. Některé z parametrů jsou dány a některé může uživatel zadat. Mezi neměnné parametry patří koeficienty $a_n,\ldots,\ a_1,\ a_0$ levé strany, řád n levé strany, koeficienty $b_m,\ldots,b_1,\ b_0$ pravé strany, řád m pravné strany. Tvar LDR je tedy zadaný pevně ve zdrojovém souboru ulohaX.jsp.

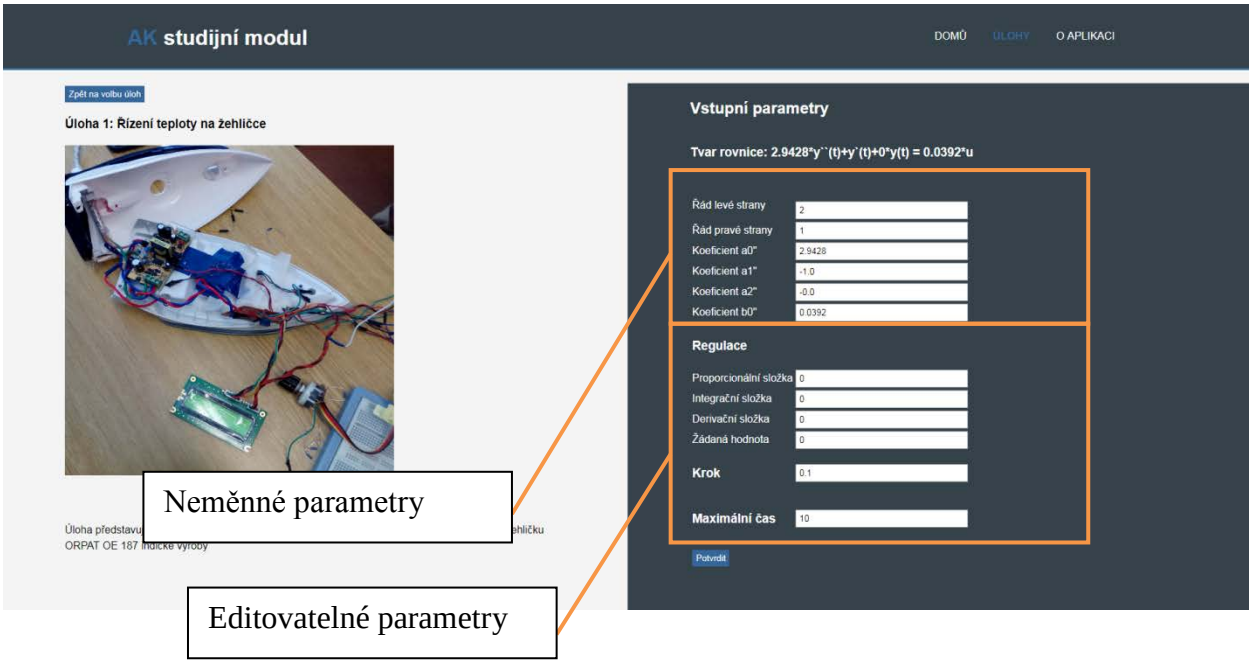
$$a_n \cdot y^{(n)}(t) + \ldots + a_1 \cdot y'(t) + a_0 \cdot y(t) = b_m \cdot u^{(m)}(t) + \ldots + b_1 \cdot u'(t) + b_0 \cdot u(t),$$

jako v kap. 2.1.

Mezi parametry, které uživatel může nastavit patří parametry regulátoru: K_R , T_I , T_D, jako v kap. 3.2.

Dále mezi nastavitelné parametry patří vstupující požadovaná hodnota $w(t)$ jako v kap. 1.

Nastavit lze také krok T numerické iterační metody Runge-Kutta a maximální čas t_{max} jako v kapitole 4.1.



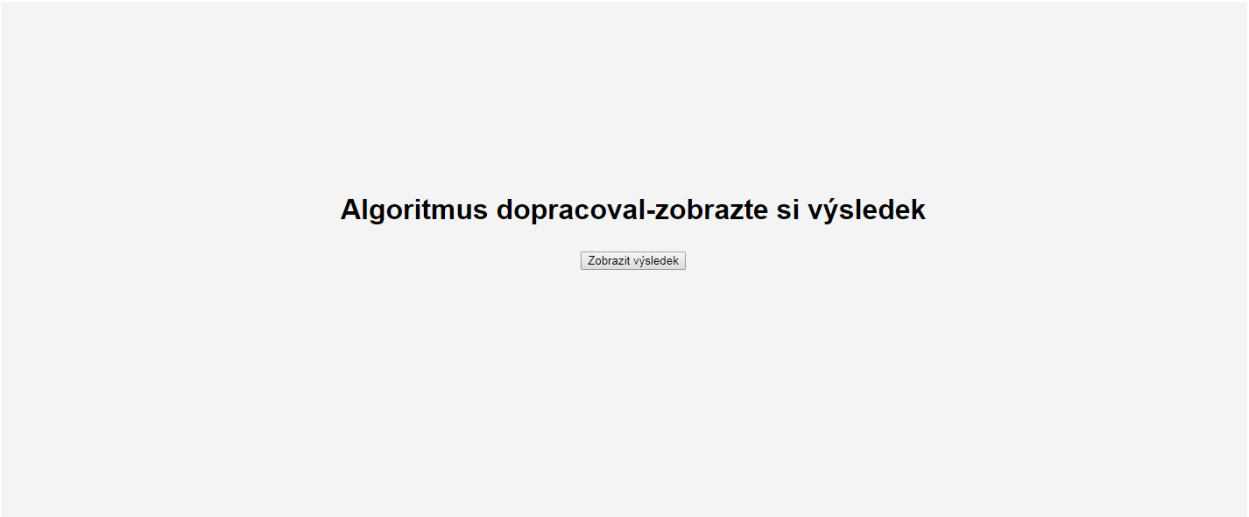
Obr. 17.: UlohaX.jsp

9.2.5 Vypocet.java

Servlet vypocet.java je komplexnější částí webové aplikace a bude popsán v kap 10. Servlet přijme neměnné parametry pro danou úlohu a editovatelné parametry, které se mění dle vstupu od uživatele jako v předchozí kapitole 9.2.4. Správnost editovatelných vstupů je ošetřena v kódu. V případě jejich nesprávnosti výpočet pokračuje na javaerr.jsp. K testování vstupů slouží následující kód v Javě, který je zapsaný v servletu vypocet.java:

```
/* testování parametru P regulátoru*/
try
        {
             konst_P_test = Double.parseDouble(konst_P_str);
        }
catch(Exception e)
        {
             konst_P_test_str = "1";
             request.setAttribute("konstTestP",konst_P_test_str);
             RequestDispatcher rd= request.getRequestDispatcher("/javaerr.jsp");
             rd.forward(request, response);
          }
```

V případě správně zadaných vstupů webová aplikace pokračuje po stisku tlačítka zobrazit výsledek jako na obr 18. na javavysledek.jsp

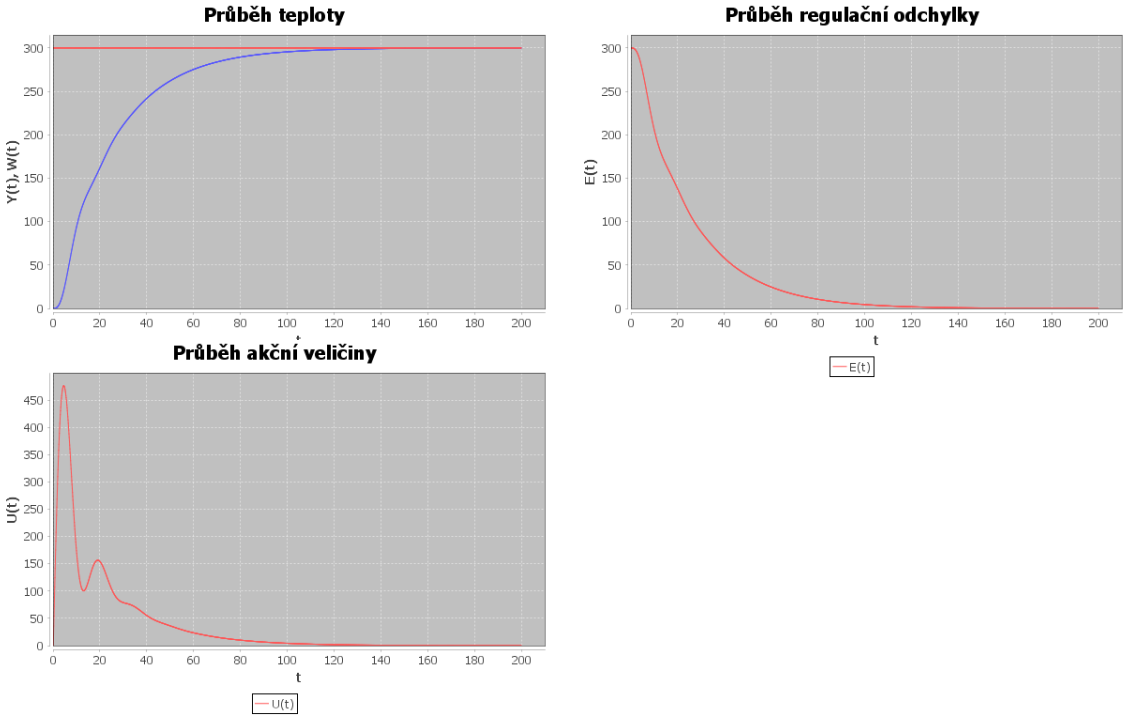


Obr. 18.: Tlačítko „zobrazit výsledek“

9.2.6 Výstup z aplikace

Výstup ze servletu je realizován prostřednictvím javavysledek.jsp.

Výstupem ze servletu jsou tři grafy závislostí $y(t) + w(t)$, $e(t)$, $u(t)$ jako na obr.19 a výčet parametrů pro přiřazení grafů vstupům.



Obr. 19.: Výstupní grafy

9.2.7 Formátování webové aplikace

Formátování webové aplikace je provedeno pomocí souboru style.css

10. ALGORITMUS PRO VÝPOČET BODŮ GRAFU

Algoritmus je obsažený v servletu vypocet.java. Hlavička metody obsažené v třídě Algrithm37 je následující:

```
public static  List<List<Double>> Algrithm37(int rad_form, int rad_mm_form,double krok,
double max_krok , int max_iterace, double leva_form[], double prava_form[],double konst_P, double
konst_I, double konst_D, double w)
```

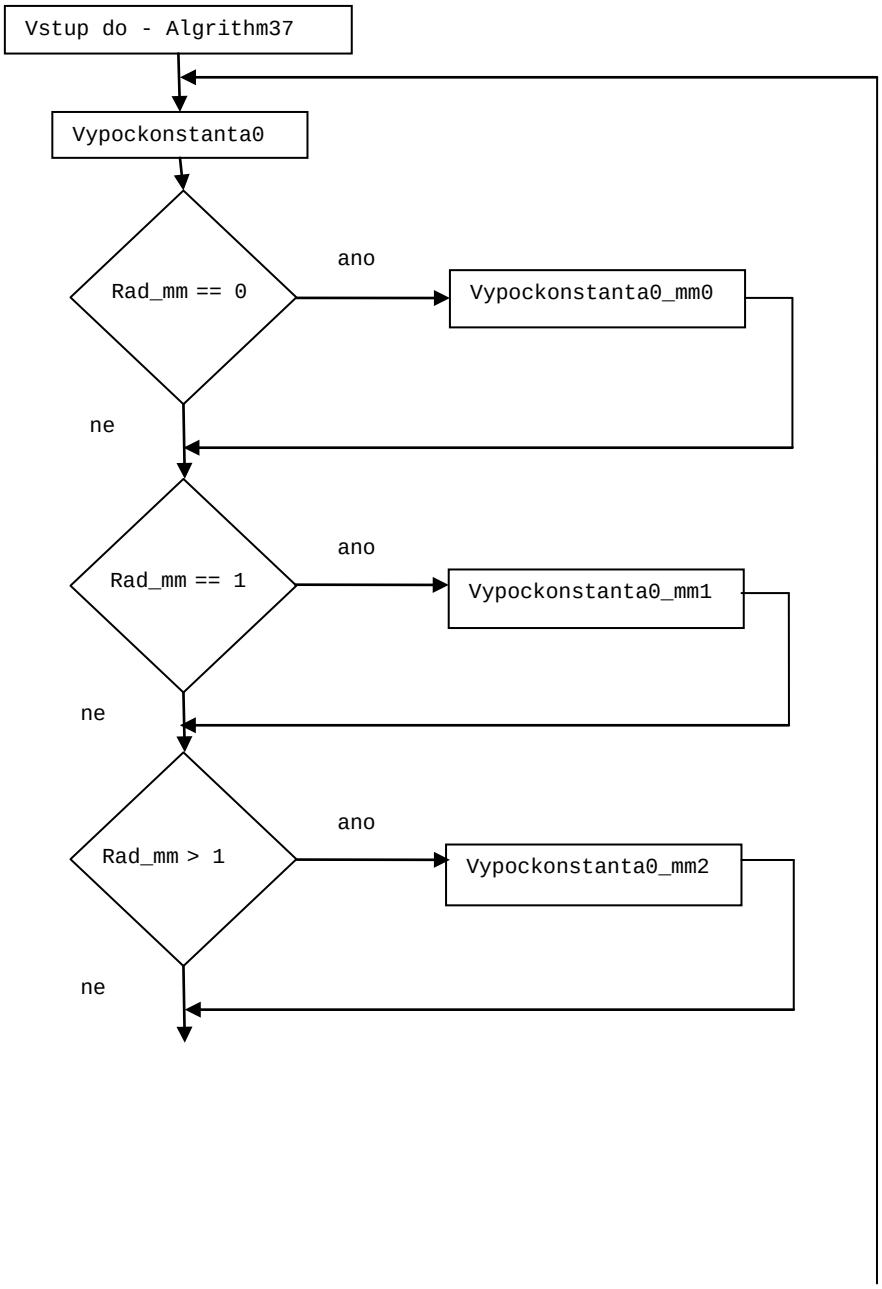
Vstupující parametry jsou `rad_form` (řád levé strany n LDR), `rad_mm_form` (řád pravé strany m LDR), `krok` (T) , `max_krok` (t_{max}), `max_iterace`(k_{max}), `leva_form[]`($a_n \dots a_1, a_0$), `prava_form[]` ($b_n \dots b_1, b_0$), `konst_P`(K_R), `konst_I`(T_I), `konst_D`(T_D) a `w`($w(t)$).

Vystupující parametry jsou typu `List<List<Double>>` a v servletu vypocet.java jsou uloženy v proměnné typu `List<List<Double>>` `VycetVsech`, ta v sobě ukládá hodnoty výstupní veličiny $y(t)$, požadované hodnoty $w(t)$, odchylky $e(t)$, akčního zásahu $u(t)$, a času t.

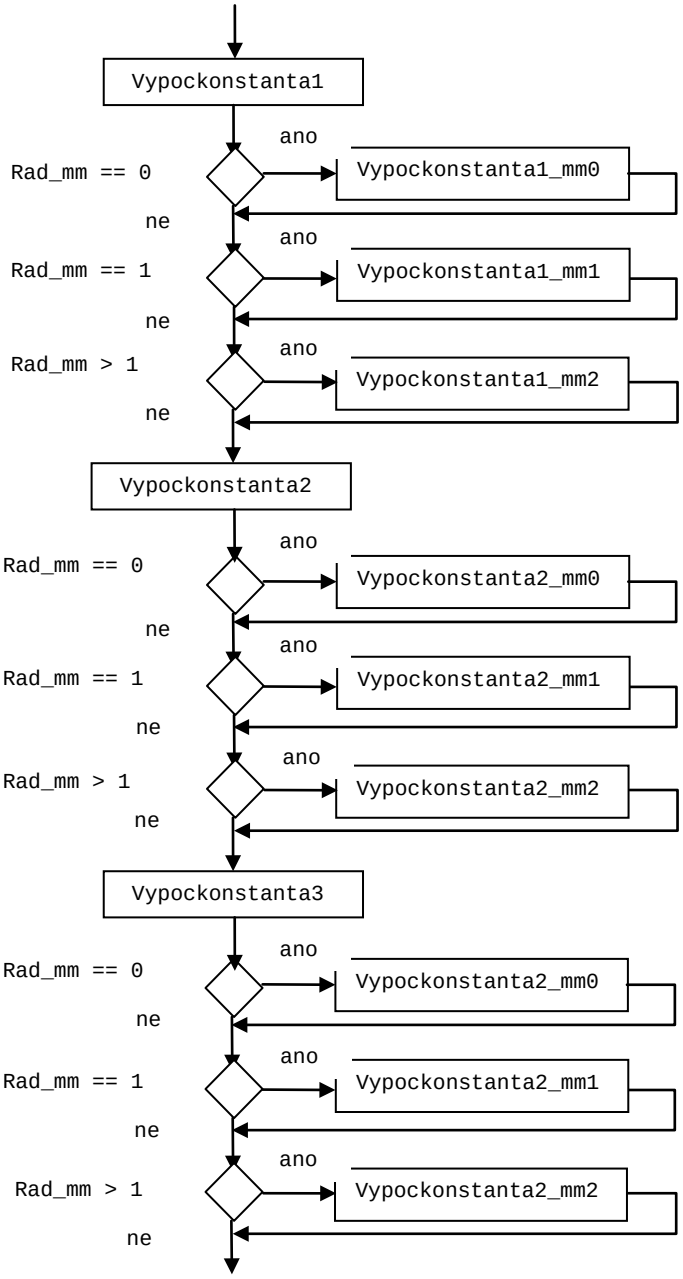
10.1 Vývojový diagram algoritmu

Algoritmus pro výpočet bodů grafu se skládá z následujících metod:

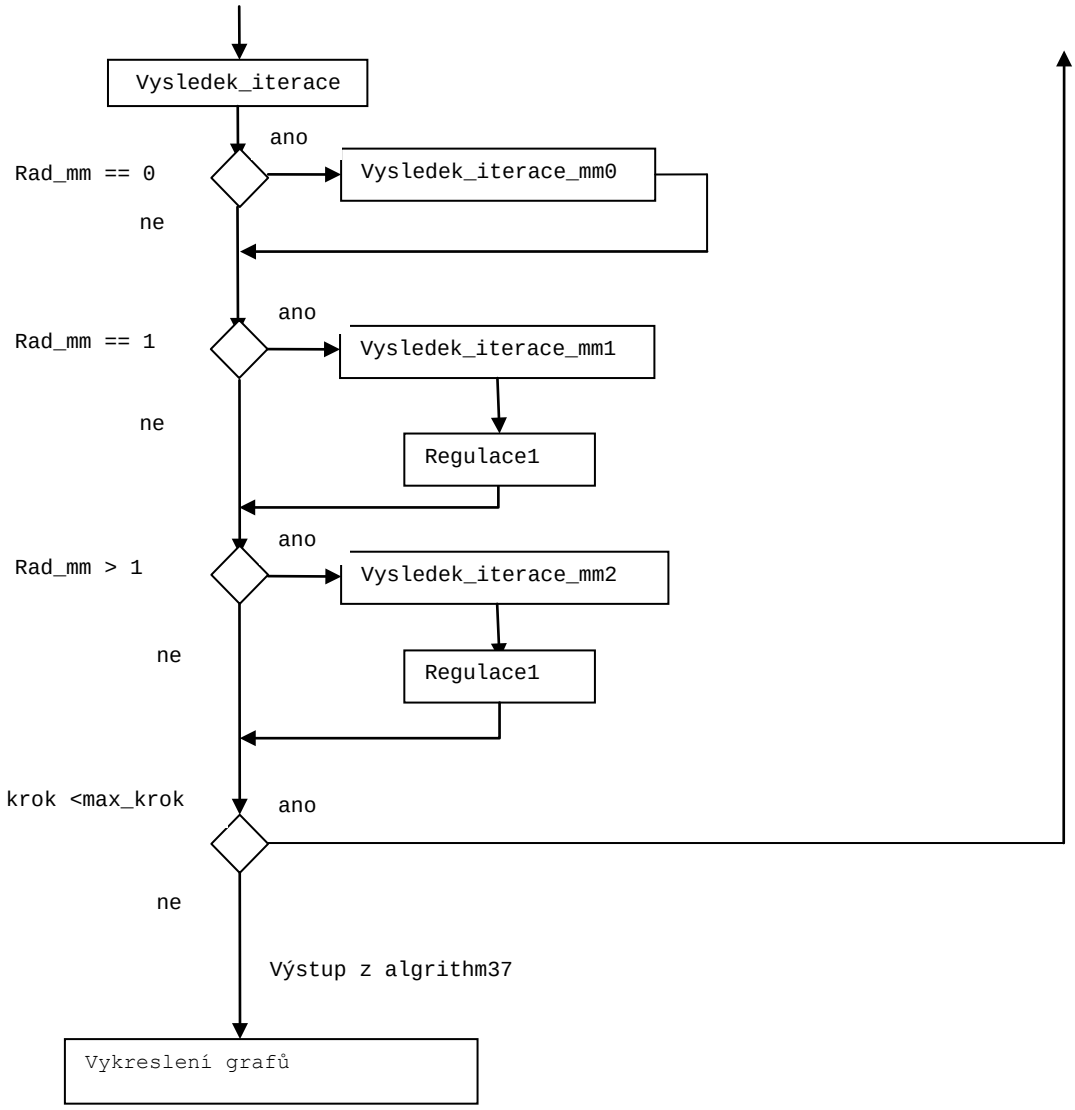
Metody `Vypockonstanta1, Vypockonstanta2, Vypockonstanta3` jsou analogické k metodě `Vypockonstanta0` s tím , že ale volají metody `VypockonstantaX_mm0, VypockonstantaX_mm1,VypockonstantaX_mm2` ,kde X je příslušné číslo uvedené na konci metody, tedy 0,1,2,3.V diagramu jsou uvedeny zjednodušeně. Metody jsou pospány dále v textu.



Obr. 20.: Vývojový diagram - část 1



Obr. 21.: Vývojový diagram-část 2



Obr. 22.: Vývojový diagram- část 3

10.2 Rozbor metod algoritmu

Metody algoritmu jsou popsány od shora dolů, tak jak jde tok informace algoritmem.

10.2.1 Metoda vypocetkonstantaX

Metody vypockonstantaX $X \in < 0,1,2,3 >$, slouží k rozzřazení do tří metod pomocí podmínek. Podmínky testují hodnotu proměnné `Rad_mm`, ta představuje řád pravé strany m LDR. Po rozřazení na `Rad_mm=0, Rad_mm=1, Rad_mm>1`

následuje volání příslušné metody `VypockonstantaX_mm0, VypockonstantaX_mm1, VypockonstantaX_mm2`, koncovka za pomlčkou např. `_mm0` představuje metodu pro `rad_mm=0` (analogicky `_mm1, _mm2).` Vše je zřejmé z vývojového diagramu viz kap 10.1. Zde uveden příklad zdrojového kódu pro `Vypockonstanta0_mm0.`

```
/*bez prave strany
  rad_mm - rozrazeni*/
if (rad_mm==0){
       pom_k =  Vypoc_Konstanta0_mm0(rad, konst_A,
       koef_leva, y);
 }
// prava strana = u
if(rad_mm == 1) {
       pom_k = Vypoc_Konstanta0_mm1(u_koncove, rad, konst_A,
       koef_prava, koef_leva, y, celkove_k, w, akt_iterace);
// prava strana u+u+‘+u‘‘...
if(rad_mm> 1){
        pom_k = Vypoc_Konstanta0_mm1more(u_koncove,rad_mm, rad, konst_A , koef_leva,
koef_prava,y, celkove_k,  w, akt_iterace);
       }
         return pom_k;
     }
```

10.2.2.Metoda vypocetkonstantaX_mm0

Metoda slouží k výpočtu konstant k_1, k_2, k_3, k_4 metody Runge Kutta 4. řádu,jako v kap. 5.2.1. Uvedený příklad zdrojového kódu je pro `Vypoc_Konstanta2_mm0` a pro $\mathbf{k_3} = \boldsymbol{f}(x_i + \frac{1}{2}\cdot h, \boldsymbol{y_i} + \frac{1}{2}\cdot h \cdot \mathbf{k_2})$.

```
public static double[][] Vypoc_Konstanta2_mm0(int  rad, double konst_A, double[]koef_leva, double
[]y, double krok, double[][]celkove_k)

{

        double sum = 0;

       int i = 0;

       int j = 0;

        double [][]pom_k = new double[4][4];

        for( j = 0; j<rad;j++) {

                for (i = 0; i< rad; i++){

                        // Tvar bez prave strany

                        if(j == 0){

                            if((i==0) && (konst_A>0 || konst_A<0)) // prisumeni konstanty na leve

                            {

                              // pro konst_A=1 představuje jednotkový skok

                             sum += konst_A;

                            }

                              sum += koef_leva[i+1]*(y[rad-i-1] +(double)(((double)1/(double)2)
              *krok*celkove_k[1][i]));

                                        // koef_leva[i+1]*(y_i[rad − i − 1] + 1/2 · h · k_2)

                            }

                         else{

                                sum = y[rad-j]+(double)(((double)1/(double)2)*krok

                               *celkove_k[1][j-1]);

                               }

                        }

                       pom_k[2][j]= sum

                      sum = 0; // vynulování sumační proměnné sum

                       }

              return pom_k;

              }
```

Proměnná část kódu pro metody vypocetkonstanta_mm1 a vypocetkonstanta_mm2

10.2.3 Metoda vypocetkonstantaX_mm1

Metoda `Vypoc_Konstanta2_mm1` se liší od `Vypoc_Konstanta2_mm0` ,tím že obsahuje proměnnou `List<List<Double>> u_koncove` ,ta je vypočtena v samém závěru algoritmu v metodě `Regulace1`. Při inicializaci je nastavena na 0. Níže je uvedena proměnná část kódu pro metodu `Vypoc_Konstanta2_mm1` z kap. 10.2.2.

```
                                        ⋮
                {
                 sum += konst_A;
                }
                sum += koef_leva[i+1]*(y[rad-1-i]+(double) (((double)1/(double)2)
                *krok*celkove_k[1][i]));
                 if(i ==0 ){
                   sum += (u_koncove.get(akt_iterace).get(0)*koef_prava[0]);
                }        //a_n · y^(n)(t) + … + a_1 · y'(t) + a_0 · y(t) = b_0 · u(t)
                }         // koef_prava[0] · u(t)
            else{
                                        ⋮
```

10.2.4 Metoda vypocetkonstantaX_mm2

Metoda `Vypoc_Konstanta2_mm2` se liší od `Vypoc_Konstanta2_mm0` ,tím že obsahuje proměnnou `List<List<Double>> u_koncove`. Níže je uvedena proměnná část kódu pro metodu `Vypoc_Konstanta2_mm2` z kap. 10.2.2.

```
                                        ⋮
              sum += konst_A;
             }
            sum += koef_leva[i+1]*(y[rad-1-i]+(double)(((double)1/(double)2)*
             krok*celkove_k[1][i]));
             if(i ==0 ){
                      sum += u_koncove.get(akt_iterace).get(0);
              }//a_n · y^(n)(t) + … + a_1 · y'(t) + a_0 · y(t) = b_m · u^(m)(t) + … + b_1 · u'(t) + b_0 · u(t)
        }
        else{

                                        ⋮
```

10.2.5 Metoda vysledek_iterace

Metoda výsledek iterace slouží k rozřazení stejně jako metoda `VypockonstantaX`. Pomocí podmínek `Rad_mm=0, Rad_mm=1, Rad_mm>1`

se program rozvětví na `Vysledek_iterace_mm0, Vysledek_iterace_mm1` a `Vysledek_iterace_mm2`.

10.2.6 Metoda vysledek_iterace_mm0

Metoda `Vysledek_iterace_mm0` slouží k výpočtu

$\boldsymbol{y}_{i+1} = y_i + \frac{1}{6} \cdot T \cdot (\boldsymbol{k_1} + 2 \cdot \boldsymbol{k_2} + 2 \cdot \boldsymbol{k_3} + \boldsymbol{k_4})$, jako v kap. 5.2.1.

```
public static List<Double> Vysledek_iterace_mm0(List<List<List<Double>>> VystupY_proGraf,int
rad,double[] y,  double krok, double [][]celkove_k, int max_iterace, int navyseni_radu, int
rad_mm)

{
             int j = 0;
             int i = 0;
             double[] pom_y = new double[rad+rad_mm];
             List<Double> y_mezi= new ArrayList<Double>();
             y_mezi.clear();
             double u=0;
             double e=0;
             for(j=0; j< rad+navyseni_radu;j++) {//y_{i+1} = y_i + 1/6 · T · (k_1 + 2 · k_2 + 2 · k_3 + k_4),
                      pom_y[j] = y[rad-j-1] + ((double)1/(double)6)*krok*
                      (celkove_k[0][j]+2*celkove_k[1][j]+2*celkove_k[2][j]+celkove_k[3][j]);
                      y_mezi.add(pom_y[j]);
             }
                      y_mezi.add(u);
                      y_mezi.add(e)
           return y_mezi;
 }
```

Proměnná část kódu pro vysledek _iterace_mm1

10.2.7 Metoda vysledek_iterace_mm1

Metoda `Vysledek_iterace_mm1` má oproti `Vysledek_iterace_mm0` navíc parametry :

`List<List<Double>> u_koncove` (hodnota $u(t)$), `double u_pred`(hodnota $u(t)$ z předchozí iterace),, `int akt_iterace`(aktuální iterace),, `double w` ($w(t)$), `double konst_P, double konst_I, double konst_D` jsou složky regulátoru V této metodě je volána metoda `Regulace1` pomocí , které se dopočítá předběžná hodnota $u(t)$), ve zdrojovém kódu uvedena jako `du` . Pro její vypočtení je nutné znát $e(t)$ jako v kap. 3.1. Pro získání konečného $u(t)$ je třeba sečíst `du` a `u_pred`. Níže je uvedena proměnná část kódu pro metodu `Vysledek_iterace_mm1` z kap. 10.2.6.

```
⋮
 double du = 0;
if(akt_iterace >0) {// pokud je iterace >0, tak již je záznam v u_koncove.get(akt_iterace).get(0)
       u_pred = u_koncove.get(akt_iterace).get(0)
}
for(j=0; j< rad+navyseni_radu;j++) {
             pom_y[j]=y[rad-j-1]+((double)1/(double)6)*krok*(celkove_k[0][j]+2*celkove_k[1][j]
                +2*celkove_k[2][j]+celkove_k[3][j]);
                y_mezi.add(pom_y[j]);
}
                e = w -pom_y[rad-1];  //výpočet e(t)
               du = Regulace1(0,pom_y, VystupY_proGraf , e,   konst_P,  konst_I,  konst_D ,
              rad,krok, akt_iterace, rad_mm); // volání metody Regulace1
                u = du +u_pred; // výpočet  u(t)
⋮
```

Proměnná část kódu pro ve vysledek _iterace_mm2

10.2.8 Metoda vysledek_iterace_mm2

Oproti `Vysledek_iterace_mm1` má metoda `Vysledek_iterace_mm2` navíc `double[] koef_prava(`představuje koeficienty pravé strany LDR $... = b_m \cdot u^{(m)}(t) + ... + b_1 \cdot u'(t) + b_0 \cdot u(t)$, tedy $b_m, ..., b_1, b_0$.). Pomocí substituce rovice $a_n \cdot y^{(n)}(t) + ... + a_1 \cdot y'(t) + a_0 \cdot y(t) = b_m \cdot u^{(m)}(t) + ... + b_1 \cdot u'(t) + b_0 \cdot u(t)$ na dvě rovnice:

$$a_n \cdot z^{(n)}(t) + ... + a_1 \cdot z'(t) + a_0 \cdot z(t) = 1 \cdot u(t)$$

$$y(t) = b_m \cdot z^{(m)}(t) + ... + b_1 \cdot z'(t) + b_0 \cdot z(t)$$

Níže je uvedena proměnná část kódu pro metodu `Vysledek_iterace_mm2` z kap. 10.2.7.:

```
for(i = 0;i<rad_mm;i++) { //y(t) = b_m · z^(m)(t) + ... + b_1 · z'(t) + b_0 · z(t)
        pom_y[rad] +=  koef_prava[rad_mm-1-i]*pom_y[rad-rad_mm+i];
}
y_mezi.add(pom_y[rad]);
```

10.2.9 Metoda regulace1

V metodě regulace1 se vypočítá proměnná du zmíněná výše v textu. Používá algoritmus řízení uvedený v kap. 4.2. Zdrojový kód metody:

```
 public static double Regulace1(int j ,double[] pom_y,List<List<List<Double>>>
yN_proGraf,double e,  double konst_P, double konst_I, double konst_D ,int rad,double krok, int
akt_iterace, int rad_mm ){
        double pom_u = 0.0;
        int i = 0;
        double xc = 0.0;// −y(k · T)
        double xc_KT1 = 0.0;// y[(k − 1) · T]
       double xc_KT2 = 0.0;// y[(k − 2) · T]
        double T = krok;
        double x = 0.0;
         x = pom_y[j];
        int promenna_y= 0.0;
        if(rad_mm >1){
               xc = pom_y[rad];
         }else{
                xc = pom_y[rad-1];
        }
         if(rad_mm > 1){
                promenna_y = rad;
         }else{
                promenna_y = rad-1;
         }
        if(akt_iterace == 1){  /* vypocet v první iteraci- nezáme xc_KT1, xc_KT2*/
         xc_KT1  =0;
        xc_KT2  = 0;
        pom_u = konst_P*((-xc)+xc_KT1 + ((T/konst_I)*e)+(konst_D/T)*((-xc)+2*xc_KT1-xc_KT2));
         }




        if(akt_iterace == 2){ /* vypocet v druhé iteraci- nezáme xc_KT2*/
                  xc_KT1  =yN_proGraf.get(promenna_y).get(akt_iterace).get(0);
```

```
            xc_KT2  = 0;
            pom_u= konst_P*((-xc)+xc_KT1 + ((T/konst_I)*e)+(konst_D/T)*((-xc)+2*xc_KT1-
            xc_KT2));
        }
        if(akt_iterace>2){
            xc_KT1  = yN_proGraf.get(promenna_y).get(akt_iterace).get(0);
            xc_KT2  = yN_proGraf.get(promenna_y).get(akt_iterace-1).get(0);
            pom_u= konst_P*((-xc)+xc_KT1 + ((T/konst_I)*e)+(konst_D/T)*((-xc)+2*xc_KT1-
           xc_KT2));
       }
        return pom_u;
}
```

11. GRAFY

Výstupní grafy jsou řešeny prostřednictví volně dostupných knihoven pro tvorbu grafů JFreeChart od Jfree.org. [15]

11.1 Vykreslení grafů

Jsou naprogramovány v souborech `XYLineChart_v16.java`, `XYLineChartU_v16.java`, `XYLineChartE_v16.java`.

```
        // volání ve vypocet.java
        XYLineChart_v16 chart = new XYLineChart_v16("Zavislost Y(t), W(t)",
        "Zavislost Y(t), W(t)",  VycetVsech, pocet_iteraci, rad_int, rad_mm_int);
        // příklad zdrojového kódu z XYLineChart_v16
        // tvorba grafu
        JFreeChart xylineChart = ChartFactory.createXYLineChart( chartTitle , "t" , "Y(t), W(t)"
,createDataset(VycetVsech, pocet_iteraci2, rad, rad_mm) ,  PlotOrientation.VERTICAL ,  true ,
true , false);{
        try { // Vytvoření a uložení obrázku na disk
                final ChartRenderingInfo info =  new ChartRenderingInfo();
                final  File file1= new File("C:\\ NetBeansProjects\\VyukovyModulAKFinalv4\\web\\
                ChartY.png");
                ChartUtilities.saveChartAsPNG(file1, xylineChart, 600,400, info);
        }
        catch(Exception e){



         }
}
```

```
public XYDataset createDataset(List<List<Double>> VycetVsech, double pocet_iteraci2, int rad, int
rad_mm) {
        YSeries w = new XYSeries( "W(t)" );
        int a;
        // rozlišení rad_mm = 0, 1 a rad_mm > 2
        if (rad_mm>= 2){
               a =0;
        }
        else{
                a= 1;
        }
        int i = 0;
        // uložení bodů vypočtených v algorithm36 do XY datasetu
        for (i = 0;i<= pocet_iteraci2; i++){
                w.add( VycetVsech.get(0).get(i) , VycetVsech.get(4+rad-a).get(i) );
         }
        String yt = "Y(t)";
        XYSeries rad_y = new XYSeries(yt);




        // uložení bodů vypočtených v algorithm36 do XY datasetu
        if(rad_mm>= 2){
                for (i = 0;i<= pocet_iteraci2; i++){
                       rad_y.add( VycetVsech.get(0).get(i) , VycetVsech.get(rad+1).get(i) );


                     }
         }
         if(rad_mm< 2){
                for (i = 0;i<= pocet_iteraci2; i++){
                       rad_y.add( VycetVsech.get(0).get(i) , VycetVsech.get(rad).get(i) );
                      }
         }
```

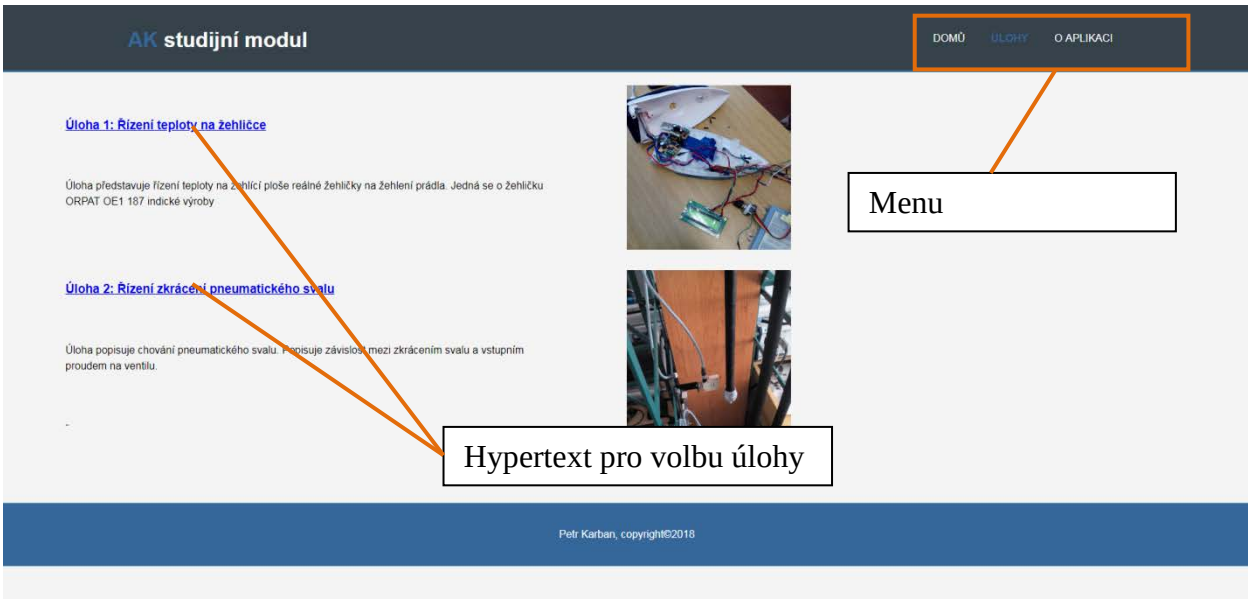
```
// přidání dat do data setu
 final XYSeriesCollection dataset = new XYSeriesCollection( );
 dataset.addSeries( w );
 dataset.addSeries( rad_y);
 return dataset;
}
```

12. UKÁZKA CHODU WEBOVÉ APLIKACE

Úlohy v aplikaci jsou dvě, s tím že je velmi jednoduché přidat další úlohy vytvořením ulohyX.jsp a editováním webové stránky na několika místech. Tento postup zde nebudu uvádět. Aplikace v tuto chvíli zpracovává dvě demonstrativní úlohy. První úloha je řízení teploty na žehličce. Druhá úloha je řízení zkrácení pneumatického svalu. Pro ukázku jsem si zvolil první úlohu , na které předvedu chování aplikace.

12.1 Volba úlohy

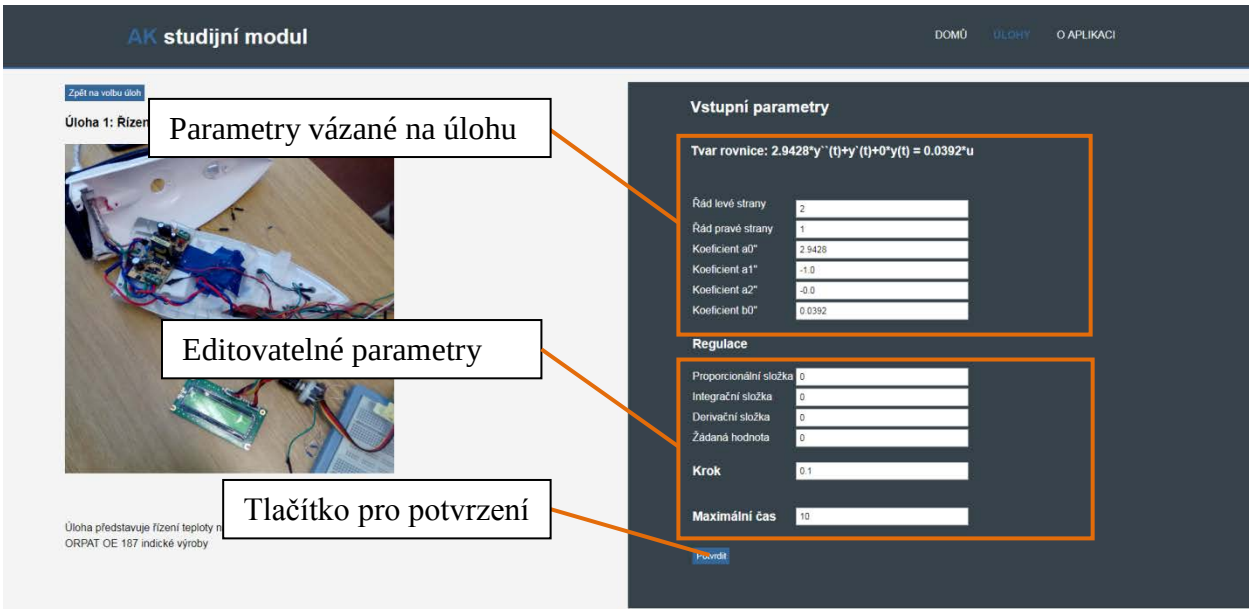
Napřed z hlavní obrazovky přejdeme pomocí menu do obrazovky pro volbu úlohy, tím že klikneme na možnost Úlohy v menu. Dále mezi hypertextovými odkazy v levém sloupci webové aplikace zvolíme první úlohu.



Obr. 23.: Volba úohy

12.2 Spuštění úlohy

Po vstupu do menu dané úlohy můžeme zkontrolovat neměnné parametry úlohy. Následně zvolíme editovatelné parametry. Až budou parametry navolené stiskneme tlačítko potvrdit v pravé dolní části obrazovky. Úloha se začne počítat.

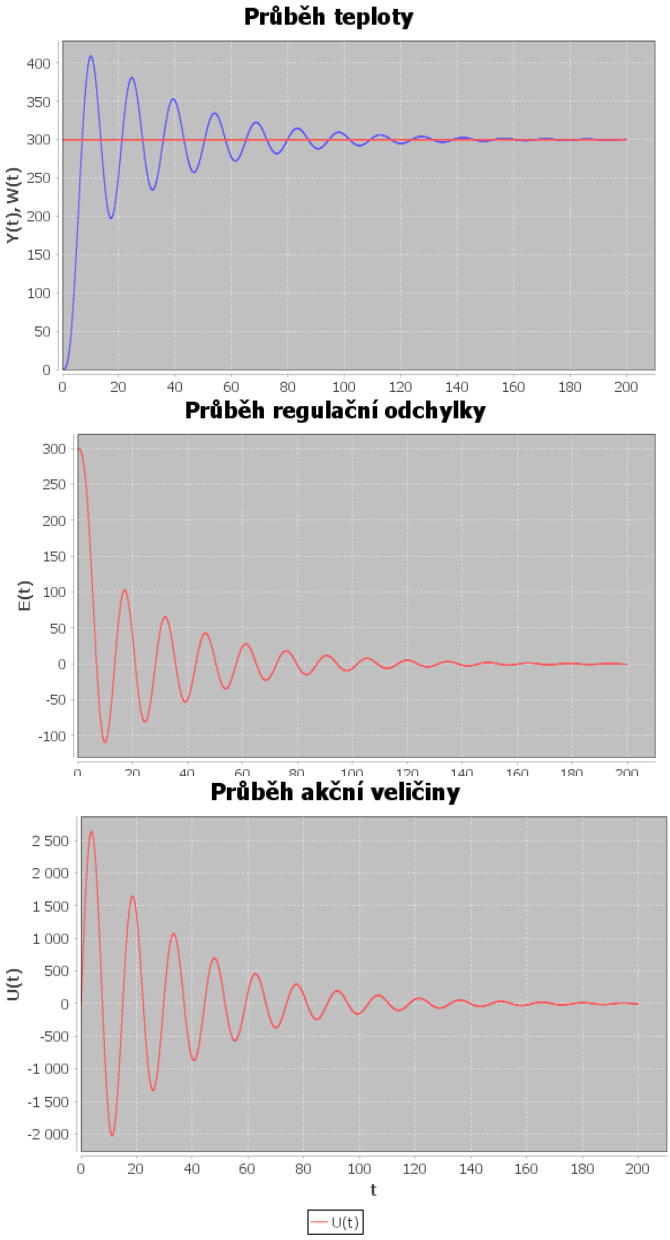


Obr. 24.: Spuštění úlohy

12.3 Výstup z úlohy

Výstupy pro nastavení první úlohy. Maximální čas t_{max} je 200. Krok je 0.1. Regulátor je nastaven na hodnotu zesílení K_R 15. Integrační časová konstanta T_I je nastavena na 4 a derivační časová konstanta T_D je nastavena na 0. Požadovaná hodnota výstupu $w(t)$ je nastavena na 300.

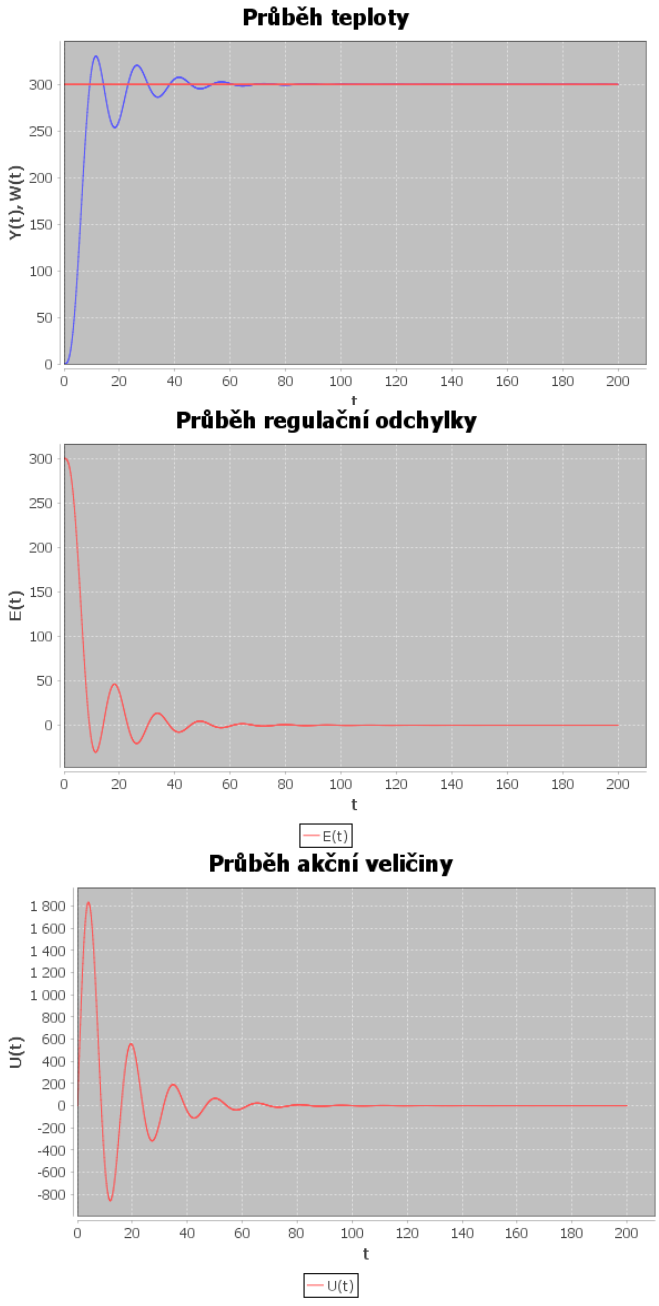
Výstup pro tyto parametry je vyobrazen na obr. 25.



Obr. 25.:Výstup z úlohy 1 – část 1

Výstupy pro nastavení první úlohy. Maximální čas t_{max} je 200. Krok je 0.1. Regulátor je nastaven na hodnotu zesílení K_R 15. Integrační časová konstanta T_I je nastavena na 6 a derivační časová konstanta T_D je nastavena na 0. Požadovaná hodnota výstupu $w(t)$ je nastavena na 300.

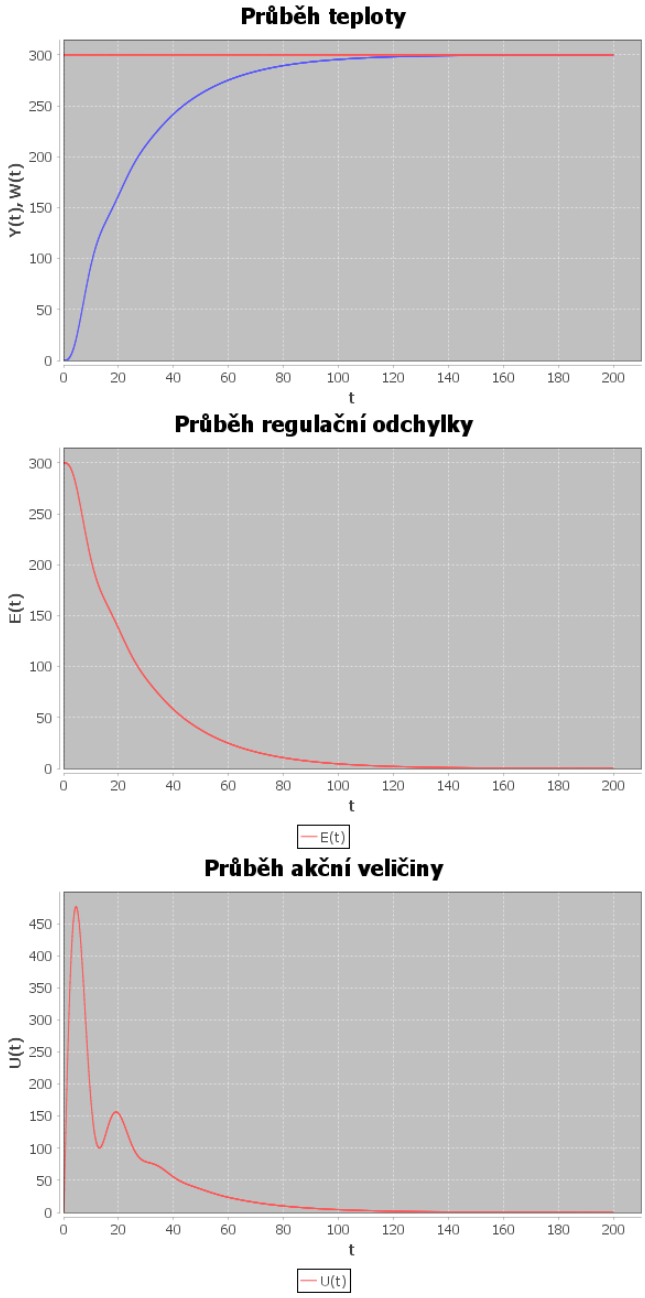
Výstup pro tyto parametry je vyobrazen na obr. 26.



Obr. 26.: Výstup z úlohy 1 - část 2

Výstupy pro nastavení první úlohy. Maximální čas t_{max} je 200. Krok je 0.1. Regulátor je nastaven na hodnotu zesílení K_R 15. Integrační časová konstanta T_I je nastavena na 25 a derivační časová konstanta T_D je nastavena na 0. Požadovaná hodnota výstupu $w(t)$ je nastavena na 300.

Výstup pro tyto parametry je vyobrazen na obr. 27.



Obr. 27.: Výstup z úlohy – část 3

ZÁVĚR

V rámci diplomové práce byla navržena a vytvořena webová aplikace s virtuálními úlohami pro podporu výuky předmětu aplikovaná kybernetika. Aplikace byla psána pomocí HTML, CSS , Java Server Pages a technologie servetů. Je v ní možné napojit spojitý PID regulátor na model popisující reálný problém(virtuální úloha). V aplikaci jsou v současné chvíli dvě virtuální úlohy: řízení teploty na žehličce a řízení zkrácení pneumatického svalu. Množství virtuálních úloh je možné úpravami ve zdrojovém kódu(duplicitou) navýšit.

Úloha má jak neměnné parametry, tak parametry které je možné zvolit. Mezi neměnné parametry patří koeficienty lineární diferenciální rovnice, pravé a levé strany rovnice. Dále pak řád pravé a levé strany rovnice. Mezi editovatelné parametry patří požadovaná výstupní hodnota ze soustavy a maximální čas po který se bude úloha počítat. Dále pak čas , který úloha „ujde“ při jednom cyklu algoritmu a nakonec pak parametry regulátoru proporcionální, integrační a derivační složka.

Webová aplikace byla testována na obou příkladech. Výstup testu z příkladu řízení teploty na žehličce je uveden v kapitole 12.3.

Aplikaci je možné dále rozšířit například o možnost výpisu bodů grafů při stisku tlačítka u grafu nebo o možnost vykreslování přechodových charakteristik přímo v aplikaci. Tuto možnost je v tuto chvíli již možné využít při testování algoritmu v prostředí Netbeans.

SEZNAM POUŽITÉ LITERATURY

[1] BALÁTĚ M. *Automatické řízení 1. vydání,* Praha: BEN-technická literatura,2003 . ISBN 80-7300- 020-2.

[2] OLEHLA, M. a S. NĚMEČEK. *Základy aplikované kybernetiky*. 2. vyd. Liberec: Technická univerzita v Liberci, 2005. ISBN 80-7083-952-X.

[3] HOFREITER M., *Základy automatického řízení(skriptum)*, Praha: České vysoké učení technické v Praze, 2012. ISBN 978-80-7372-297-5.

[4] BALÁTĚ M. *Vybrané statě z automatického řízení,* 2.vyd. Vysoké učení technické v Brně, 1996 . ISBN 80-214- 0793-X.

[5] FAJMON, B. a I RŮŽIČKOVÁ. *Matematika 3* [online]. Brno: UMAT FEKT, 2004 [cit. 2018-30-04]. Dostupné z: http://www.umel.feec.vutbr.cz/VIT/images/pdf/ studijni_materialy/bc/Matematika3.pdf

[6] WAGNEROVÁ R.,MINÁR K. *Prezentační a výukový modul pro oblast analýzy regulačních obvodů v prostředí Intranetu* [online] [cit. 2018-30-04]. Dostupné z: http://books.fs.vsb.cz/Analyza/

[7] SERAFÍN Č. *Stabilita diskrétního regulačního obvodu* [online]. [cit. 2018-30-04]. Dostupné z: http://www.kteiv.upol.cz/index.php?page=mechatronika

[8] SCHLEGEL M. *Průmyslové PID regulátory: Teorie pro praxi*[online]. [cit. 2018-30-04]. Dostupné z: http://zcu.arcao.com/kky/zky/Prago1.pdf

[9] DUŠEK F. *, Matlab a Simulink : úvod do používání,* Univerzita Pardubice, 2000, ISBN 80-7194- 273-1.

[10] NETBEANS.ORG.[online]. [cit. 2018-30-04]. Dostupné z: https://netbeans.org/features/index.html#

[11] APACHE.ORG[online] [cit. 2018-30-04].

Dostupné z: http://tomcat.apache.org/

[12]W3SCHOOLS[online] [cit. 2018-30-04].

Dostupné z: https://www.w3schools.com/html/html_intro.asp

[13]W3SCHOOLS[online] [cit. 2018-30-04].

Dostupné z: https://www.w3schools.com/css/default.asp

[14] LORENC P., *Dynamické generování obsahu s Java Server Pages*, Brno, 2003, Bakalářská práce, Masarykova univerzita, fakulta informatiky.

Dostupné z: https://is.muni.cz/th/139661/fi_b/bc_-_Petr_Lorenc.pdf

[15]JFREE.ORG[online] [cit. 2018-30-04].

Dostupné z: http://www.jfree.org/index.html

SEZNAM OBRÁZKŮ

Obr. 1.: Princip regulace 13
Obr. 2.: Blokové schéma systému 14
Obr. 3.: Jednotkový skok vstupující do systému 15
Obr. 4.: Přechodová charakteristika 15
Obr. 5.: Blokové schéma regulátoru 16
Obr. 6.: Blokové schéma P-I-D regulátoru 18
Obr.7.: Přechodová charakteristika P-I-D regulátoru 18
Obr. 8.: Příklady průběhů pro P-regulátor-statická soustava 19
Obr. 9.: Dosažení nulové odchylky 20
Obr. 10.: Vzorkování 22
Obr. 11.: Lineární nekmitavý průběh 1. řádu - stabilita 26
Obr. 12.: Kmitavé průběhy - stabilita 27
Obr. 13.: Znázornění oblasti stability 28
Obr. 14.: Schéma webové aplikace 35
Obr. 15.: Index.jsp 36
Obr. 16.: Ulohy.jsp 37
Obr. 17.: UlohaX.jsp 39
Obr. 18.: Tlačítko „zobrazit výsledek“ 40
Obr. 19.: Výstupní grafy 41
Obr. 20.: Vývojový diagram - část 1 43
Obr. 21.: Vývojový diagram-část 2 44
Obr. 22.: Vývojový diagram- část 3 45
Obr. 23.: Volba úohy 57
Obr. 24.: Spuštění úlohy 58
Obr. 25.:Výstup z úlohy 1 – část 1 59
Obr. 26.: Výstup z úlohy 1 - část 2 60
Obr. 27.: Výstup z úlohy – část 3 61

SEZNAM TABULEK

Tab.1.: Pravidla pro určení parametrů 30

PŘÍLOHA

Jako příloha je vloženo CD, které obsahuje:

- Diplomovou práci v elektronické podobě v souboru DP.pdf