UNIVERZITA PARDUBICE

Fakulta ekonomicko-správní

DIPLOMOVÁ PRÁCE

2008 **Petr GALČÍK**

Univerzita Pardubice

Fakulta ekonomicko-správní

Využití aplikace WebServer pro výuku

Petr Galčík

..

Diplomová práce

2008

Univerzita Pardubice
Fakulta ekonomicko-správní
Ústav systémového inženýrství a informatiky
Akademický rok: **2007/2008**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Petr GALČÍK**
Studijní program:	**M6209 Systémové inženýrství a informatika**
Studijní obor:	**Informatika ve veřejné správě**
Název tématu:	**Využití aplikace WebServer pro výuku**

Zásady pro vypracování:

Předpokládá se, že diplomová práce bude obsahovat:
- popis aplikace MATLAB Web server, jeho součástí a funkcí,
- popis vytvoření aplikací s využitím Web server-u,
- řešení vybrané školní úlohy pomocí funkcí aplikace MATLAB Web server.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

KARBAN, P.: *Výpočty a simulace v programech Matlab a Simulink*. **1. vydání. Brno: CPress, 2006. 224 s. ISBN 80-251-1301-9.**

THE MATHWORKS, Inc.: *Matlab Web Server* **[online]. [cit. 2007-10-11]<http://www.mathworks.com/products/webserver>.**

THE MATHWORKS, Inc.: *Matlab - The Language of technical computing*. **[online]. [cit. 2007-10-11]<http://www.mathworks.com/...>.**

ZAPLATÍLEK, K.: *Matlab - tvorba uživatelských aplikací*. **1. vydání. Praha: Ben, 2004. 216 s. ISBN 80-7300-133-0.**

Vedoucí diplomové práce: **doc. Ing. Jiří Křupka, Ph.D.**
Ústav systémového inženýrství a informatiky

Datum zadání diplomové práce: **22. října 2007**

Termín odevzdání diplomové práce: **26. května 2008**

L.S.

doc. Ing. Renáta Myšková, Ph.D.
děkanka

doc. Ing. Pavel Petr, Ph.D.
vedoucí ústavu

V Pardubicích dne 22. října 2007

Poděkování:

Děkuji především panu doc. Ing. Jiřímu Křupkovi, Ph.D za odborné vedení, cenné rady a připomínky k diplomové práci a hlavně za trpělivost, se kterou se mi věnoval při tvorbě diplomové práce. Dále bych rád poděkoval svým rodičům za celoživotní podporu a za to, že mi umožnili studovat.

Souhrn

Tato diplomová práce se zabývá vytvořením internetové aplikace, která umožní on-line řešení vybraných úloh z předmětu Teorie systémů. Každá úloha MWS aplikace nejprve obsahuje teoretický výklad následovaný praktickým příkladem. MWS aplikace využívá možností MATLAB Web Serveru. Dále práce popisuje strukturu webových stránek MWS aplikace a orientaci v těchto stránkách.

Klíčová slova:

HTML; MATALB Web Server; Teorie systémů.

Title:

Usage WebServer application for education

Abstract

This graduation theses describe process of creation an Internet application, which enables on-line solution some problems from subject Theory of system. Every problem first includes theoretic followed practical exercise. The MWS application makes use of MATLAB Web Server capabilities. Then work describes structure web pages of MWS application and orientation in these pages.

Keywords

HTML; MATLAB Web Server; System theory.

Obsah

ÚVOD 1

1 TEORETICKÝ POPIS MATLAB WEB SERVER APLIKACE 2

1.1 Jednotlivé části MWS aplikace 2

1.2 MWS aplikace jako systém 3

1.3 Aplikace MWS z pohledu e-learningu 6

2 MATLAB WEB SERVER 8

2.1 Prostředí MATLAB Web Serveru 8

2.2 Apache HTTP server 9

2.3 Součásti MATLAB Web Serveru 10
2.3.1 Soubor matlabserver.conf 11
2.3.2 Program matweb.exe 11
2.3.3 Soubor matweb.m 11
2.3.4 Soubor matweb.conf 11

2.4 Princip fungování MATLAB Web Serveru 12

3 VYTVOŘENÍ MWS APLIKACE 15

3.1 Vstupní formulář 15
3.1.1 Kontrola zadaných údajů JavaScriptem 16

3.2 Vytvoření m-souboru 18
3.2.1 Formát dat 19
3.2.2 Kontrola zadaných údajů MATLABem 19

3.3 Výstupní formulář 20

4 PŘÍKLADY ŘEŠENÝCH ÚLOH 23

4.1 Jednoduché regulátory 23
4.1.1 Jednotlivé typy regulátorů 23
4.1.2 Příklad v MWS aplikaci 25

4.2 Model hospodářské regulace 26
4.2.1 Matematický popis modelu multiplikátoru-akcelerátoru 27
4.2.2 Stabilizační politika 28
4.2.3 Příklad v MWS aplikaci 29

4.3 Stabilita regulačních obvodů 30
4.3.1 Stabilita spojitých systémů 30
4.3.2 Příklad v MWS aplikaci 33

4.4 Řízení s korekčními členy 35
4.4.1 Druhy korekcí 35

4.4.2 Příklad v MWS aplikaci 37

5 STRUKTURA WEBOVÝCH STRÁNEK MWS APLIKACE 39

5.1 Struktura webových stránek MWS aplikace 39

5.2 Struktura vstupního formuláře MWS aplikace 40

5.3 Mapa webových stránek MWS aplikace 41

6 PROBLÉMY PŘI TVORBĚ MWS APLIKACE 42

ZÁVĚR 44

POUŽITÁ LITERATURA 46

SEZNÁM OBRÁZKŮ 48

SEZNAM TABULEK 48

SEZNAM PŘÍLOH 49

Úvod

V současnosti, kdy informační a komunikační technologie mají stále významnější pozice ve vzdělávacím procesu, jsou kladeny stále nové požadavky na klasické i nové formy vzdělávání. Veškeré tyto požadavky přinášejí změny vzdělávacího procesu hlavně ve formě a v komunikaci mezi studenty a pedagogy a umožňují vnímání nových poznatků způsoby, které ve srovnání s tradičními formami vzdělávání umožňují dosáhnout vyšší efektivity ve vzdělávání. Jednou z možností je e-learning. Rozvoj e-learningu je umožněn rychlým rozšiřováním Internetu.

Cílem této práce je vytvoření e-learningové aplikace, kterou je možné využít při výuce. Pomocí této aplikace je možné řešit prostřednictvím Internetu vybrané úlohy, spadající do oblasti předmětu Teorie systémů. Jedním z nástrojů, umožňující vytvoření této aplikace, může být MATLAB, konkrétně toolbox MATLAB Web Server (dále MWS). Aplikaci, která bude vytvořena, se bude nazývat MATLAB Web Server aplikace (dále MWS aplikace). MWS poskytuje možnost komunikace mezi MATLABem a uživatelem prostřednictvím Internetu. Samotná MWS aplikace se nachází na serveru a je ovládána uživatelem pomocí HTML stránek.

První kapitola se věnuje teoretickému popisu MWS aplikace, jeho částí, popisu MWS aplikace jako systému a na závěr kapitoly jsou popsány prostředky e-learningu a proces zavádění e-learningového systému.

Ve druhé kapitole je vysvětleno, jakým způsobem MWS může pracovat a jaké jsou jeho části a co je třeba k nastavení MWS a jak přesně funguje.

Třetí kapitola popisuje vytvoření jednotlivých částí MWS aplikace. K tomu, aby MWS fungoval, je nejdříve nutné vysvětlit nastavení APACHE HTTP serveru. Po nastavení serveru je popsáno vytvoření vstupního a výstupního formuláře v HTML kódu a m-soubor v MATLABu včetně možností, jak ošetřit chyby při zadávání údajů ze strany uživatelů.

Čtvrtá kapitola obsahuje teoretický výklad k vybraným úlohám z předmětu Teorie systémů a vychází. V každé podkapitole je po teoretickém výkladu uveden příklad z MWS aplikace.

V páté kapitole je popsána struktura vstupních a výstupních HTML stránek včetně mapy webových stránek MWS aplikace.

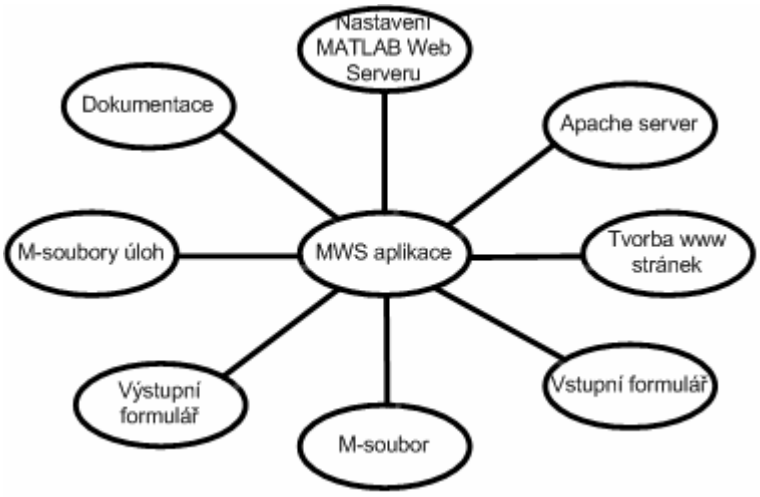
V šesté kapitole jsou uvedeny nejčastější chyby, objevující se při vytváření MWS aplikace nejčastěji a možné řešení k jejich odstranění.

1 Teoretický popis MATLAB Web Server aplikace

V této kapitole budou popsány části, které jsou součástí MWS aplikace. Dále bude objasněn způsob, jakým MWS aplikace pracuje. Na MWS aplikaci lze také nahlížet jako na systém, což umožní lepší popis prvků systémů a procesů, které v systému mezi jeho prvky probíhají. Dále se musí určit, na jaké architektuře systém MWS aplikace pracuje a které funkce by aplikace měla aplikace uživatelům poskytovat.

1.1 Jednotlivé části MWS aplikace

MWS aplikace se skládá z několika částí, které na sebe vzájemně navazují a každá z nich má určitou úlohu, nezbytnou pro činnost MWS aplikace. Každý prvek MWS aplikace zastupuje jeden krok při její tvorbě a zahrnují např. nastavení HTTP serveru či konfiguraci MATLAB Web Serveru. Jaké prvky MWS aplikace zahrnuje, je na obr. 1.1 a pořadí jednotlivých prvků v pravém směru ukazuje proces tvorby a nastavení MWS aplikace.



Obr. 1.1: Součásti MWS aplikace. Zdroj: vlastní.

Každá část je podrobně popsána v dalších částech této práce. Prvotním krokem tvorby MWS aplikace je *nastavení MWS*, aby komunikace mezi uživatelem a MATLABem prostřednictvím www stránek probíhala korektně a bez problémů. Druhým krokem je konfigurace *Apache serveru*, který zajišťuje vzájemnou komunikaci. Samotná tvorba www stránek zahrnuje následující prvky *vstupní formulář* a *výstupní formulář*. *M-soubor* a *m-soubory úloh* jsou prvky pracující s daty, zadané uživateli a provádějící činnosti, pro které byly vytvořeny. Závěrečným krokem je vytvoření dokumentace.

1.2 MWS aplikace jako systém

Schéma MWS aplikace je na obr.1.2. Podle tohoto modelu je systém rozdělen na lokální a vzdálený počítač. Na lokálním počítači pracuje uživatel, zatímco vzdálený počítač zajišťuje ty požadavky, které jsou na systém tohoto typu kladeny. Systém obsahuje prvky zajišťující činnost systému, např. HTTP server, výpočetní systém MATLAB, MATLAB Web Server, Simulink.[9]

V lokálním počítači jsou prostřednictvím webových stránek zadávány hodnoty pro řešené příklady a po provedení výpočtů jsou výsledky opět zobrazeny prostřednictvím webových stránek opět na straně lokálního počítače. Nutnou podmínkou pro komunikaci mezi lokálním a vzdáleným počítačem je, aby počítače byly vzájemně propojeny prostřednictvím počítačové sítě. Toto je řešeno prostřednictvím Internetu, protože se předpokládá široká dostupnost MWS aplikace nejen během vyučovacích hodin. Komunikace využívá uživatelské WWW rozhraní a probíhá mezi lokálním a vzdáleným počítačem. Je založena na architektuře klient – server, ve které si klienti a servery nejsou rovnocenní a mají jasně rozdělené úlohy. Samotná komunikace mezi lokálním a vzdáleným počítačem začíná připojením k webovým stránkám aplikace a probíhá na základě HTTP protokolů který definuje soubor pravidel pro přístup k souborům různého charakteru a pro přenos informací. [10]

Lokální systém obsahuje pouze vstup do systému, což je uživatel, který se prostřednictvím webového prohlížeče přihlásí k webovým stránkám MWS aplikace. Uživatel je kromě zadávaných hodnot vstupním prvkem systému, který si prostřednictvím webového prohlížeče zobrazuje webové stránky ve formátu HTML. [9] Může si jednotlivé úlohy zobrazit a k nim provést výpočty či simulace, které využívají SIMULINKu, nebo si prostudovat teoretický výklad k úlohám.

Vzdálený počítač obsahuje prvky zajišťující činnost systému. Jak je uvedeno, komunikace využívá protokolu HTTP. Komunikaci zajišťuje HTTP serveru, který je umístěn ve vzdáleném počítači. V tomto systému se využívá služeb APACHE Serveru.

Takto navržený model lze rozdělit do tří obecných částí:

- Vstupní formulář (webová stránka),
- M-soubor,
- Výstupní formulář (opět webová stránka).

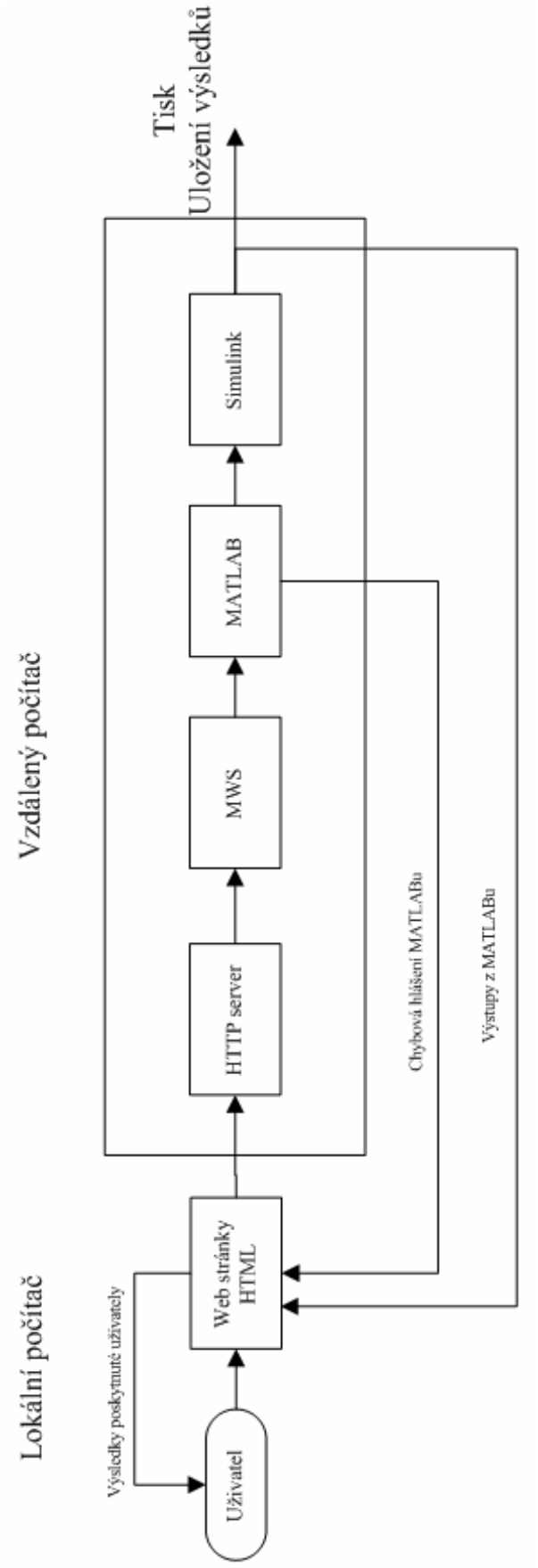
Po zvolení příkladu uživatel vloží hodnoty ve vstupním formuláři a prostřednictvím HTTP serveru se data odešlou z lokálního počítače do MWS, který je potřebný pro komunikaci s výpočetním systémem MATLAB a uživateli. MWS běží jako služba operačního

systému. Data jsou prostřednictvím MATLABu odeslány konkrétnímu m-souboru, pomocí něhož lze i spouštět simulační modely vytvořené v SIMULINKu. Výsledky, které poskytne SIMULINK jsou ve výstupním formuláři zobrazeny uživateli. Pokud je s výsledkem spokojen, může si tyto výsledky vytisknout či si je uložit. [9]

Výsledky výpočtů mohou být:

- Číselné údaje,
- Obrázky nebo grafy vygenerované Malabem a uložené v běžném grafickém formátu (JPEG, MBP apod.),
- Smíšený výstup (grafika i text).

Uživateli jsou pomocí webových stránek zobrazeny výsledky, které jsou poskytnuty MATLABEM. Uživateli se tyto situace poskytují prostřednictvím zpětné vazby z webových stránek.



Obr. 1.2: Struktura systému MWS aplikace. Zdroj: vlastní.

1.3 Aplikace MWS z pohledu e-learningu

E-learning je vzdělávací proces, který využívá počítače a jim blízká zařízení ke zprostředkování výukových a tréninkových materiálů (v užším smyslu) a k podpoře procesu vzdělávání (v širším smyslu). Takto uvedená definice je spíše obecným vymezením pojmu e-learning. Na e-learning lze také nahlížet jako na [3,7]:

- Systém - e-learning je všeobsahující termín obecně užívaný ve vztahu k počítačově zdokonalenému vzdělávaní.
- Prostředek - e-learning je výuka s využitím informačních technologií a internetu. Je formou vzdělávaní, který využívá multimediální prvky. Lze jej tedy považovat za elektronickou podporu výuky a představuje využití jednotlivých e-learningových aktivit ve vzdělávání.
- Zdroj informací – e-learning je v podstatě využívání elektronických prostředků k dosažení vzdělávacího cíle s tím, že je realizován prostřednictvím počítačových sítí.
- Proces – e-learning je proces, který využívá informačních a komunikačních technologií k tvorbě kurzů, k distribuci studijního obsahu a komunikaci mezi studenty a pedagogy.

Cílem e-learningu je maximální využití všech zdrojů a prostředků při učení. Avšak hlavním cílem je minimalizace času studenta při učení a tím snížit finanční prostředky potřebné pro studium. Bylo prokázáno, že studenti si nejlépe uchovávají znalosti, které si sami mohou vyzkoušet. K dalším důvodům, proč je dobré zavádět e-learning do výuky, může patřit snížení nákladů na klasické vzdělávání. E-learningové vzdělávání je časově nezávislé a individuální studium. Studenti si mohou zvolit dobu, kdy se budou vzdělávat. Využitím e-learningu je také zajištěna vysoká úroveň předávaných informací.

E-learning, jak již bylo zmíněno, využívá informačních technologií. K technickým prostředkům patří využívání osobních počítačů a jeho periferií. Počítače mají také zahrnovat hardwarové komponenty, které jsou nutné pro tvorbu kurzů. K softwarovým prostředkům patří aplikace, které vytvářejí e-learningové kurzy. Při výběrů aplikací potřebných pro tvorbu kurzů je nutné zohlednit, jaké nároky jsou na jednotlivé kurzy kladeny.

Ke stěžejním částem e-learningových kurzů, které jsou využívány pro potřeby výuky, jsou doplňující úkoly či elektronické kurzy a mají sloužit jako doplňky ke studiu. K cílům může například patřit naučit studenta hledat různá řešení, aplikovat při řešení úkolů určité pojmy či respektovat jisté postupy pro řešení úkolů (navazuje na text studijního článku).

Dalším úkolem e-learningu může být zdokonalení dovedností studenta či naučit studenta odpovědnosti za své řešení. [7]

Zavádění e-learningového systému lze rozdělit do pěti základních, na sebe navazujících kroků [3]:

- Prvotním krokem bývá zřízení serveru, na kterém jsou k dispozici elektronické materiály, zpravidla v podobě HTML dokumentů. Jedná se o statické zdroje, jejichž hlavní předností je okamžitá dostupnost odkudkoliv. Z pohledu tvůrců se jedná o relativně rychlou implementaci.
- Druhým krokem je statický obsah kurzů doplněn o mechanismy znalostního managementu, které umožní lépe obsluhovat, třídit, prohledávat a propojit je do tzv. Báze digitálních vzdělávacích objektů. Potud se e-learning zaobírá pouze zdroji informací a znalostí, které mají být zpřístupněny, ale už se nezabývá procesem vzdělávání.
- Ve třetím kroku jsou do e-learningových systémů začleněny dynamické prvky, které umožní vzdělávací proces rozložit na kroky a podúlohy.
- Po rozčlenění vzdělávacích procesů do kroků se lze zaměřit směrem k uživateli. E-learningový systém, vypracovaný až k této úrovni umožňuje rozpoznávat potřeby studentů, k nimž je pak možné definovat učební cíle.
- V poslední fázi zavádění e-learningu je začleněn do ostatních procesů. Zde již vystupuje jako aplikační vrstva znalostního managementu.

Navrhovaná MWS aplikace umožňuje provázání elektronických materiálů a MWS. Studenti dostávají možnost si vyzkoušet chovaní daných systémů či modifikovat jejich parametry. Studenti se dostávají z pasivních rolí, kdy pouze studují určité příklady popisující chování systémů, do role aktivně pracujících, kdy si podle zájmu mohou modelovat příklady na dané témata. Pokud výsledky nejsou podle očekávání, či neodpovídají tomu, jak to student pochopil, může se vrátit k teoretickým výkladům.

2 MATLAB Web Server

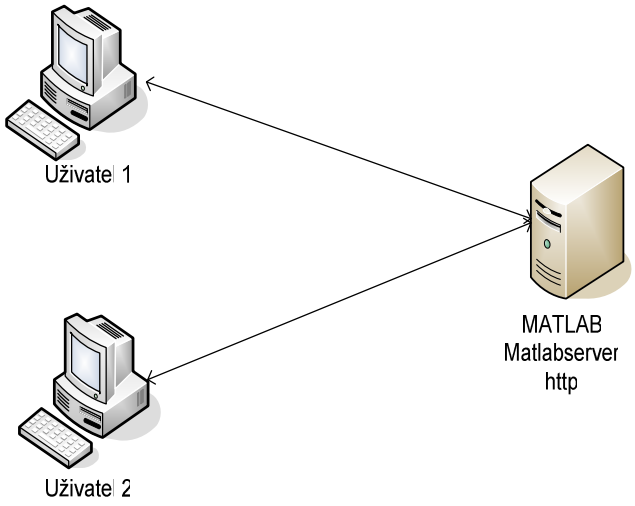
Toolbox MATLAB Web Server byl standardně volitelnou součástí instalace programu MATLAB až do verze 7.1. Ve verzi MATLAB R2007 již tento toolbox není podporován a byl nahrazen podporou JavaScript a .NET FRAMEWORK. Přesto jsou jeho možnosti využití při vytváření různých aplikací poměrně široké. Mezi hlavní výhodu MWS patří jeho jednoduchost, kdy MWS aplikaci by měl být schopen každý, kdo má základní znalosti programování. Před nastavením MWS je nutné nastavit HTTP server, který zajišťuje komunikaci mezi uživatelem a MWS.

2.1 Prostředí MATLAB Web Serveru

MWS umožňuje vytvářet MWS aplikace a jsou schopné rozhraním World Wide Web zasílat data do MATLAB-u k požadovaným výpočtům a výsledky zobrazí ve webovém prohlížeči. MWS je závislý na protokolu TCP/IP pro přenos dat mezi klientským systémem a Malabem. Požadovaný síťový software a hardware musí být instalován na systému, který je prioritně používán MWS. [14]

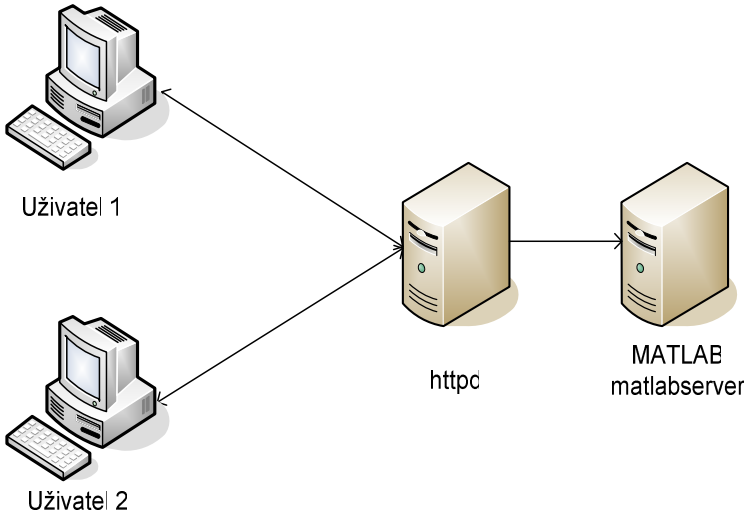
Jsou ale nutné ještě další podmínky, které musí být splněny pro správnou činnost *MWS*. Je tedy nutné, aby byl k dispozici HTTP server podporující CGI (Common Gateway Interface). V našem případě je využit Apache HTTP Server, který slouží k vytváření a správu serverů.

V nejjednodušším případě pracuje web server na klientské stanici, zatímco MATLAB, MWS a Web démon (http) pracují na jiném počítači (serveru). [14]



Obr. 2.1: HTTP server a MWS na jednom počítači. Zdroj: The Mathworks[14], s.14.

Ve složitějších sítích Web démon může pracovat odděleně od ostatních prostředků. Tento systém odpovídá detailní struktuře, ve které je každá činnost podrobně popsána. [14]



Obr. 2.2: server a MWS každý na jiném počítači. Zdroj: The Mathworks[14], s.15.

2.2 Apache HTTP server

Apache HTTP Server je program, jenž slouží k vytvoření a správě webových serverů. Jedná se o Open Source Softwarem, a je provozován pod mnoha modifikacemi. Jeho základní verze je ke stažení na adrese http://www.apache.org.

Prvním krokem pro činnost APACHE je spuštění instalačního programu, pomocí kterého se nainstaluje Apache HTTP Server. Pro správnou činnost serveru v součinnosti s MWS je nutné učinit některé změny v konfiguračním souboru *httpd.conf*, který se nachází v instalačním adresáři serveru `\apache\conf\httpd.conf`. Je tedy potřeba změnit tyto části konfiguračního souboru [14]:

- ServerAdmin – emailová adresa správce serveru,
- ServerName – IP počítače, resp. localhost. Záleží to na tom, kde se server nachází,
- DocumentRoot – cesta k adresáři, ve kterém jsou uloženy HTML stránky MWS aplikace,
- Directory – cesta k adresáři, a jsou v něm uloženy HTML stránky MWS aplikace.

Pro správnou činnost serveru postačuje změnit pouze tyto parametry. Po změně parametrů v konfiguračním souboru je nutné server restartovat, aby se mohli projevit provedené změny.

Pro zobrazení webových stránek nyní postačí, aby uživatel zadal *ServerName*, na kterém je Apache HTTP Server umístěn. Jestliže se v prohlížeči zobrazí obsah adresáře zada-

ný v konfiguračním souboru *httpd.conf*, pracuje server správně. Je ale nutné, aby se adresy *DocumentRoot* a *Directory* shodovaly.

Pro ověření správné funkčnosti serveru stačí dodržet následující postup. Do adresáře, který je určen pro HTML prezentaci, se zkopíruje obsah adresáře *wsdemos*, jenž je součástí instalace MATLABu. Potom zkopírujeme klienta *matweb.exe*, také součástí instalace MATLABu, do adresáře `\apache\cgi-bin\`, kde jsou uloženy aplikace a jsou serverem spouštěny při požadavku na odesílání klientovi.

2.3 Součásti MATLAB Web Serveru

MWS je složen z několika částí, které jsou nutné pro jeho činnost [14]:

- *matlabserver*: řídí komunikaci mezi internetovou aplikací a MATLABem.

  Matlabserver je vícevláknový TCP/IP server. Spouští programy (M-soubor) v MATLAB-u specifikované skrytým polem mlmfile, která je obsažena v HTML dokumentu. *Matlabserver* spustí soubor matweb.m, což je požadovaný m-soubor.

  Matlabserver může být nastaven na jakýkoliv TCP/IP port, který je nastaven v souboru *matlabserver.conf*. Počet současně běžících MATLABů může být také nastaven v *maltabserver.conf*.

- *matweb*: je TCP/IP klient. Tento program využívá CGI k extrahování dat z HTML dokumentu a jejich přenosu do programu *matlabserver*-u.
- *matweb.m*: volaný m-soubor, který chce internetová aplikace spustit.

Následující dva konfigurační soubory jsou využívány ve spojení s MWS programy [14]:

- *matweb.conf*: konfigurační soubor potřebuje *matweb* pro spojení s *matlabserver*. Každá MWS aplikace musí být uvedena v *matweb.conf*.
- *hosts.conf*: pokud je tento soubor vytvořen, pouze počítače uvedené zde se mohou připojit k MWS. Názvy serverů jsou uváděny jmény, nikoliv pomocí IP adres. Operační systém přeloží názvu serveru na správnou IP adresu.

Umístění některých souborů spolupracující s webovými aplikacemi, včetně *matweb.m* musí být uvedeny v adresáři, který je uveden v konfiguračním souboru HTTP serveru. Soubory *matweb* a *matweb.conf* musí být uvedeny v /cgi-bin aliasu v konfiguračním adresáři HTTP serveru. Veškeré grafické výstupy musí být umístěny v adresáři, kam je HTTP server umisťuje a aplikace je tam mohou zapisovat. [14]

2.3.1 Soubor matlabserver.conf

Při spuštění *matlabserver* se nastaví podle nastavení, které je uvedeno v konfiguračním souboru *matlabserver.conf*. Tento soubor je vytvořen v adresáři `<matlab>/webserver`. Parametry nastavení jsou:

- **Číslo portu**, na kterém *matlabserver* poslouchá. Defaultní hodnota je nastavena na hodnotu 8888,
- **Počet současně spuštěných MATLABů**. Zde je defaultní hodnota nastavena na hodnotu 1.
- **Časový limit**. Tento parametr udává, jak dlouho (v sekundách) má čekat na spuštění *matlabserver*u.

Mezi parametrem nastavení a jeho hodnotou je nutné udělat mezeru. Pokud matlabserver nenalezne matlabserver.conf, pracuje podle defaultního nastavení. [14]

2.3.2 Program matweb.exe

Matweb je TCP/IP klient využívající Common Gateway Interface (CGI) k obdržení dat z HTML formulářů. Přenese údaje programu *matlabserver*, jenž poté spustí aplikace napsané v m-souborech. Pro přístup HTTP serveru musí být kopie programu *matweb.exe* umístěna v adresáři ukazující na alias /cgi-bin. [14]

2.3.3 Soubor matweb.m

Spuštěný soubor volaný *matlabserverem*. Do vstupního formuláře se vloží skryté pole *mlmfile*, kde jeho hodnotou je název volaného m-souboru. Pomocí tohoto pole matlabserver zavolá tento m-soubor a zajistí jeho spuštění. Následují ukázka vybrané části HTML kódu zajistí prostřednictvím programu *matlabserver* spuštění souboru `prikald.m`. [14]

```
<input type="hidden" name="mlmfile" value="priklad">
```

2.3.4 Soubor matweb.conf

V tomto konfiguračním souboru je seznam všech úloh MWS aplikace. Pokud by tyto úlohy v tomto souboru nebyly uvedeny, program *matlabserver* tyto úlohy nezná a nemohl by je dále posílat přímo do MATLABu.

K připojení k *matlabserver* program *matweb.exe* požaduje informace uložené v konfiguračním souboru *matweb.conf*. Tento soubor je umístěn ve stejném adresáři jako *matweb.exe*, tj. v adresáři odkazující na alias /cgi-bin.

Příklad *matweb.conf*:

```
[priklad1]
mlserver=localhost
mldir=C:/MWork

[priklad2]
mlserver=localhost
mldir=C:/Mwork
```

Konfigurace všech aplikací musí být ve stejném souboru. Každá proměnná se objeví na novém řádku, za kterou následuje rovnítko =, za kterým je přiřazena hodnota, např. `mlserver=localhost`. Název úlohy je uveden v hranatých závorkách [], např. `[priklad1]`. Za názvem následují všechny ostatní proměnné, jako je jméno serveru či adresa umístnění požadovaného m-souboru. V konfiguračním souboru matweb.conf je možné nastavovat tyto položky [14]:

- mldir (vyžadováno) – název aplikace MATLABu;
- mllog (optimální) – pracovní adresář pro čtení a zápis souborů. Pokud je specifikován, je tento adresář automaticky přidán do cesty MATLABu;
- mlserver (vyžadováno) – jméno serveru, na kterém matlabserver běží;
- mlport (vyžadováno) - číslo portu pro *matlabserver*. Tato hodnota musí odpovídat hodnotě nastavené v souboru *matlabserver.conf*;
- mltimeout (optimální) - čas čekání na *matlabserver* před vypršením času.

Po vytvoření nové *MWS* aplikace a zaznamenání v konfiguračním souboru *matweb.conf*, je třeba restartovat *matlabserver*. [14]

2.4 Princip fungování MATLAB Web Serveru

Nežli je samotná aplikace připravena k používání, je nutné, aby tvůrce připravil následující části systému:

- Stránky HTML s formulářem k zadávání hodnot pro výpočty či simulace.
- M-soubor v MATLABu pro vlastní výpočty.
  Parametry tohoto m-souboru jsou vstupní hodnoty z HTML formulář. Výstupem je textový řetězec či grafický výstup (kap. 1.2).
- Výstupní stránku v HTML pro jednoduché generování výstupů z MATLABu.

Samotný princip předávání informací mezi webovým prohlížečem a aplikacemi na straně serveru je založen na rozhraní CGI (*Common Gateway Interface*). Vytváří rozhraní me-

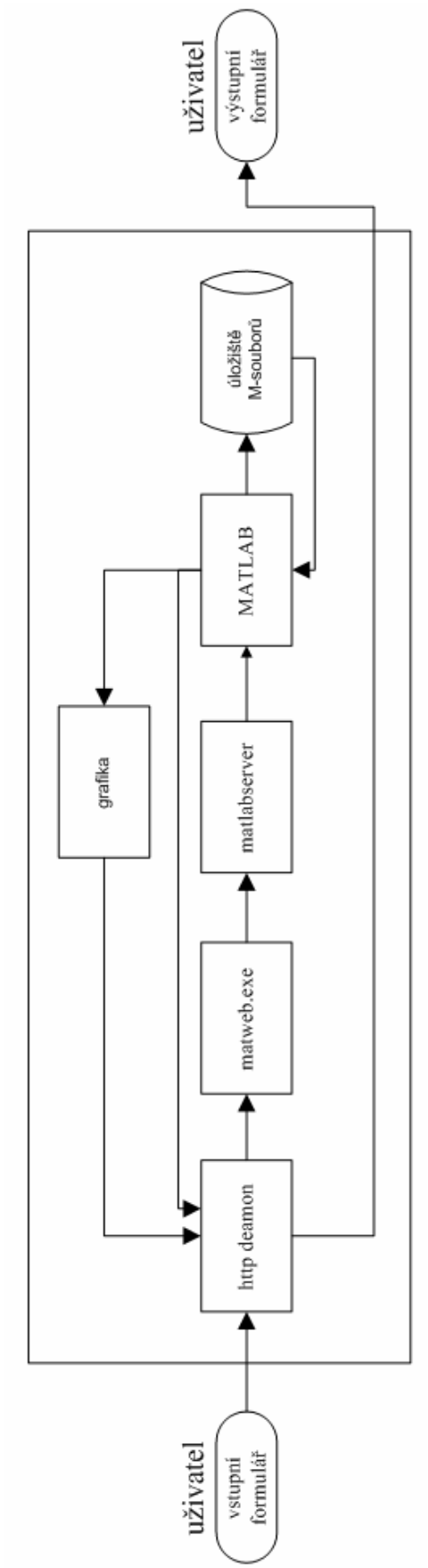
zi serverem a příslušným programem, který zpracovává data, jež převzal od WWW klienta. Je to tedy nástroj serveru, pomocí kterého je předáván příkaz od klienta ke zpracování dalšímu aplikačnímu programu (jde obvykle o skripty, tj. program napsán speciálním programovacím prostředkem). Výsledek zpracování je následně odeslán klientovi WWW jako odpověď na jeho příkaz . [10]

Samotný proces fungování aplikace je následující. Na vstupní stránce s formulářem jsou uživatelem zadány vstupní údaje, které mají být odeslány. Hodnoty jsou pomocí rozhraní CGI metodou POST odeslány programu *matweb.exe* na HTTP server. Části HTML kódu týkající se těchto činností, vypadají následujícím způsobem [14]:

```
<form action="cgi-bin/matweb.exe" method="POST">
<input type="hidden" name="mlmfile" value="priklad">
...
</form>
```

Kromě zadaných údajů se pomocí skrytého pole *mlmfile* přenese i jméno m-souboru, s nímž se má MWS spojit. Programem *matweb.exe* se přijaté údaje transformují a poté pomocí MWS jsou data zaslána odpovídajícímu m-souboru.

Již spuštěný m-soubor provede požadované činnosti a vytvoří údaje, které se mají objevit ve výstupní HTML stránce. Pro to, aby výstup byl prováděn co nejjednodušeji, je v MWS k dispozici funkce *htmlrep*. Vstupem této funkce je HTML stránka, na níž se mají výsledky zobrazit a právě výstupní hodnoty, jež byly v m-souboru vypočítány a určeny pro další prezentaci. Výsledkem funkce *htmlrep* je výstupní kód HTML stránky. Výstupní údaje jsou po ukončení činnosti m-souboru vráceny zpět přes HTTP server do WWW prohlížeče na straně uživatele. Na obr.2.3 je zobrazen princip činnost MWS. [14]



Obr. 2.3: MATAB Web Server - princip fungování. Zdroj: The Mathworks,[14], s. 51.

3 Vytvoření MWS aplikace

V této kapitole budou popsány jednotlivé kroky, které jsou třeba pro to, aby MWS aplikace byla plně funkční a plnila úkoly, pro které je vytvářena. Samotné vytvoření aplikace lze rozdělit do dvou fází.

První fází by měla být analýza úlohy, která by měla být zpracována jako MWS aplikace. Je nutné definovat typ vstupních dat a jaký typ dat uživatel zadává a co požaduje na výstupu, tedy co přesně má být výstupem z MWS aplikace (grafy, text, obrázky apod.).

Ve druhé již bude vytvářena samotná MWS aplikace. Během tohoto procesu se jako první vytvoří vstupní HTML stránka, kde se bude nacházet vstupní formulář pro zadávání dat. Ve druhém kroku se vytvoří funkce potřebné pro MATLAB (m-soubory), které vykonávají soubor příkazů v něm zapsaných, přičemž celý výpočet se realizuje prostřednictvím MWS. Třetím a závěrečným krokem je vytvoření výstupní HTML stránky, kde se zobrazují výsledky odeslané z MATLABu.

3.1 Vstupní formulář

Vstupní formulář je klasickou webovou stránkou v HTML formátu. Z tohoto formuláře jsou získávány data, která jsou následně použita m-souborem pro výpočty. Deklarace vstupního formuláře je následující:

```
<html>
<form action="cgi-bin/matweb.exe" method="POST">
<input type="hidden" name="mlmfile" value="priklad">
<!—část HTML kódu, kde se zadávají vstupní data-->
</form>
</html>
```

Vstupní formulář je označen párovým tagem `<form>…</form>`, sloužící pro získávání dat od uživatele. Párový tag FORM má následující parametry [8]:

- Action – určuje adresu, anebo cestu k programu, v našem případě k *matlab.exe*, který se spustí po odeslání údajů zadaných ve formuláři.
- Method – určuje způsob zasílání dat. Existují dvě metody GET a POST. V našem případě se využívá metoda POST.

Do vstupního formuláře se musí vložit skryté pole se jménem *mlmfile*. Tato položka určuje, který m-soubor se má spustit. V níže uvedeném příkladě bude tedy prostřednictvím MWS volán m-soubor PRIKLAD.

```
<input type="hidden" name="mlmfile" value="priklad">
```

Na závěr vstupního formuláře přidáme tlačítko k odeslání formuláře:

```
<input type="submit" value="vypocitat">
```

[14]

3.1.1 Kontrola zadaných údajů JavaScriptem

Pro správný formát dat je nezbytné, aby ve vstupním formuláři byla provedena kontrola vstupních dat. Pro kontrolu správných dat je využit JavaScript. JavaScript je programovací jazyk, jehož kód se vkládá do webových stránek. Tyto příkazy se vykonávají na straně klienta, nikoli na straně serveru. Používá se především pro úpravu stránek. [8]

Kód skriptu je možné vložit jako část stránky, nebo jako soubor s příponou .js. Skripty, které mají být volány pouze jednou, tj. při načítání stránky, je vhodné vložit do hlavičky HTML stránky mezi tagy <head>…</head>. Pro vložení JavaScriptu do stránek se používá tag :

```
<script type="text/javascript" language="JavaScript">… </script>
```

Ke kontrole údajů v MWS aplikaci je použit následující skript vytvořený v JavaScriptu.

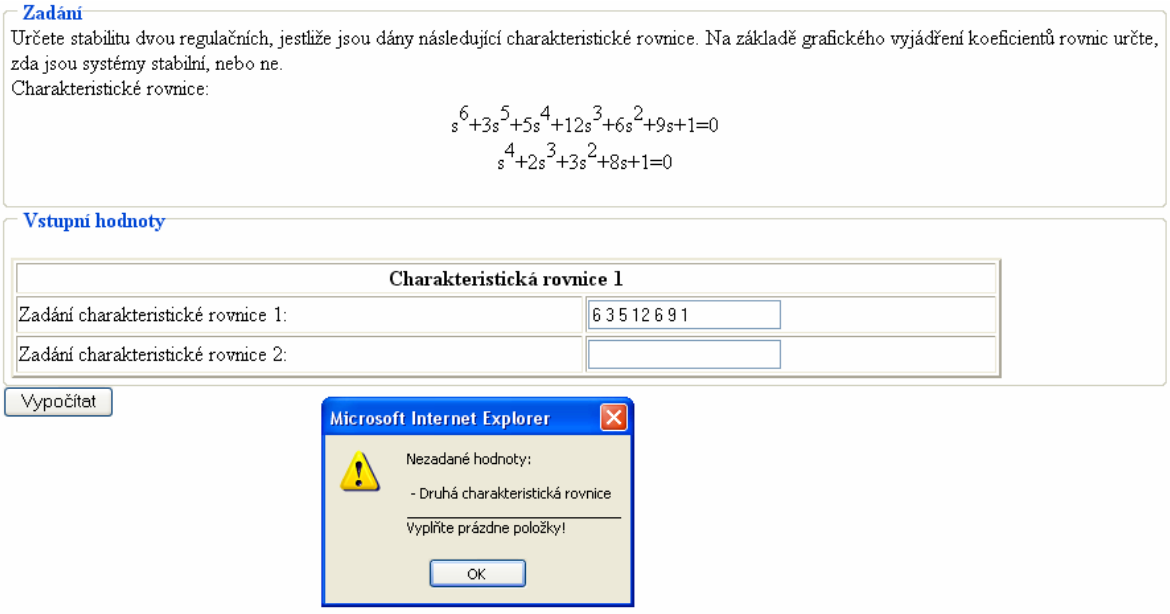
```
<script type="text/javascript" langure="JavaScript">
<!—-
function isblank(s)
{
for(var i = 0; i < s.length; i++) {
    var c = s.charAt(i);
    if ((c != ' ') && (c != '\n') && (c != '\t')) return false;
    }
return true;
}
function kontrola(formular)
{
    missinginfo = "";
    if ((formular.p1.value == "") || isblank(formular.p1.value)) {
    missinginfo += "\n – první proměnná";
    }
    if ((formular.p2.value == "") || isblank(formular.p2.value)) {
    missinginfo += "\n – druhá proměnná";
    }
    if ((formular.doba.value == "") || isblank(formular.doba.value)) {
    missinginfo += "\n – Doba simulace";
    }
```

```
if (missinginfo != "") {
missinginfo = "Nezadané hodnoty:\n" +
missinginfo + "\n_____" + "\nVyplňte prázdné položky!";
alert(missinginfo);
return false;
}
else return true;
}
-->
</script>
```

Uvedený skript má na začátku funkci *isblanc(s)*, která postupně kontroluje správnost vyplněných položek formuláře po stisknutí tlačítka „Vypočítat“. Pokud se v položkách vyskytuje např. mezera, nebo se formulář odešle bez vyplnění položky, nastane chyba. Funkce *isblanc(s)* je využita ve funkci *kontrola (formular)*, kdy jakmile je položka prázdná nebo nevyhovuje podmínkám funkce *isblanc(s)*, webový prohlížeč vypíše výstražné okno s názvem nevyplněných položek, tzv. *missinginfo* (obr. 3.1). Pokud je kontrola formuláře bezchybná, skript formuláře pokračuje dále. [11]

Ve formuláři je následně nutné určit, jakým způsobem se má spustit kontrola údajů JavaScriptem. Ve formuláři k tomuto slouží událost *onsubmit*, která je zapsána v tagu *FORM*.

```
<form onsubmit="return Kontrola();">
```



Obr. 3.1: Hlášení o chybě při nevyplnění položky ve formuláři. Zdroj: vlastní.

3.2 Vytvoření m-souboru

M-soubor zpracovává data na straně serveru a je to posloupnost příkazů pro MATLAB. M-soubor vytvořený pro MWS aplikaci musí být schopný převzít údaje ze vstupního formuláře, které následně zpracuje a odešle výstupní hodnoty do výstupního formuláře.
Všeobecná struktura m-souboru [14]:

```
function HTMLout = M-soubor(instruct,outfile)
cd (instruct.mldir); % nastavení cesty pracovního adresáře
mlid = instruct.mlid; % získání jedinečného řetězce
wscleanup('*.jpeg',0.2); % vymazání grafů
% Načítání dat ze vstupního formuláře
promenna1 = instruct.promenna1;
promenna2 = instruct.promenna2;
% Převod údajů z formuláře na data zpracovatelné Matlabem
promenna1 = str2num(promenna1);
promenna2 = str2num(promenna2);
% Všeobecné příkazy pro výpočet či simulace
% Výstupní data pro výstupní formulář
outfile.promenna1 = promenna1;
outfile.promenna2 = promenna2;
% Načtení výstupního formuláře
templatefile = which('vystup.html');
% Vygenerování výstupního formuláře
HTMLout = htmlrep(outfile, templatefile);
```

Proměnné, které jsou získané ze vstupního formuláře, jsou uloženy ve struktuře *instruct*. Jména těchto proměnných odpovídají atributům *name*, které jsou zadané ve vstupním formuláři. Po zpracování se výsledky výpočtů, názvy vygenerovaných obrázků apod. uloží do výstupní strukturované proměnné *outfile*. M-soubor končí zavoláním funkce *htmlrep*. Výstupními parametry funkce *htmlrep* jsou výstupní proměnné a výstupní formulář. Ještě před touto funkcí se do proměnné *outfile* uloží jméno výstupního formuláře pomocí funkce *which*. [14]

Základními funkcemi MATLAB Web Serveru při vytváření MWS aplikace jsou [14]:

- htmlrep – nahradí názvy proměnných ve výstupním formuláři jejich hodnotami získaných z MATLAB Web Serveru.
- wscelanup – vymaže staré soubory v určeném adresáři.
- wsprintjpeg – vytvoří a uloží graf ve formátu jpeg.

V případě simulací pro MWS aplikaci je vhodné využít Simulink. Simulinkovou schému je možné ovládat pomocí příkazů MATLABu, které jsou zapsány v m-souboru. základními použitými příkazy jsou [14]:

- *open_system('model')* – pomocí příkazu se otevře požadovaný model.
- *set_param('model/nazev_bloku','parametr1','hodnota1',…)* – příkazem nastavíme parametry jednotlivých bloků.
- *sim('model','simulacni_parametry')* – spuštění simulace podle zadaných parametrů.

3.2.1 Formát dat

Při práci se vstupními údaji je nutné vědět, jaký bude používán formát dat. Není možné zpracovat řetězce jako číslo a není možné čísla odesílat jako textové řetězce. Je nutné si uvědomit, že údaje ze vstupní stránky se načítají jako textové řetězce, přičemž nezáleží na tom, že uživatel do vstupního formuláře zadával čísla. Pro výpočet v MATLABu jsou potřeba číselné údaje, kdy jednotlivé textové řetězce se musí na ně převést. [11] K převodu textových řetězců do číselných hodnot slouží funkce *str2num*:

```
%%% Převod textového řetězce na číslo
promenna1 = str2num(promenna1);
```

Pokud jsou data odesílána zpět do výstupního formuláře, jsou číselné údaje převedeny na textový řetězec použitím funkce *num2str*:

```
%%% Převod čísla na textový řetězec
promenna1 = num2str(promenna1);
```

Rovněž pro Simulink se jednotlivé parametry bloků zadávají jako textové řetězce a je důležité použít správný formát těchto dat. Příkaz *sprintf* vloží místo proměnné *%s* textový řetězec *num2str(vektor)*.

```
%%% Převod a vložení čísla jako textového řetězce
set_param('model/blok','parametr',num2str(premenna1));
%%% Vložení vektoru jako textového řetězce
set_param('model/blok','parametr',sprintf('[%s]',num2str(vektor)));
```

Příkaz sprintf vloží místo proměnné *%s* textový řetězec *num2str(vektor)*. [11]

3.2.2 Kontrola zadaných údajů MATLABem

Podobně jako JavaScript je možné vstupní data kontrolovat i přímo MATLABem. Toto je možné využít zejména v případě, že JavaScript je ve webovém prohlížeči vypnutý. Pro

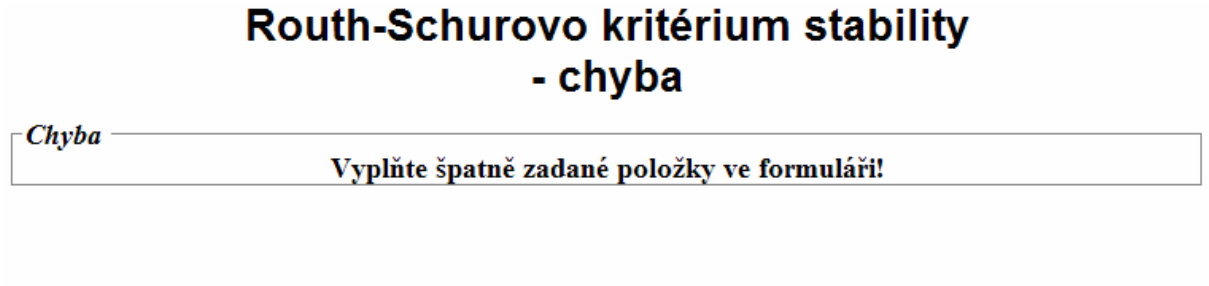
kontrolu zadaných údajů, tj. zda byly vyplněny potřebné položky vstupního formulář můžeme do příslušného m-souboru, před načítání dat ze vstupního formuláře, zařadit následující kód.

```
function HTMLout = priklad.m(in,out)
imdir = 'C:/MWork/images';
cd (in.mldir);
mlid = in.mlid;
wscleanup('*.jpeg',0.2,imdir);
errorcode=0;
%%% Kontrola zadaných parametrů
if ( isempty(in.promenna1) | isempty(in.promenna2))
 out.errormsg='Vyplňte špatně zadané položky ve formuláři !';
 errorcode=1;
end
```

Na konec m-souboru je nutné přidat informace o tom, který výstupní formulář má být použitý v případě chyby ve vstupním formuláři.

```
%%% Zobrazení chybového hlášení MATLABu při nevyplnění všech povinných
položek
templatefile = which('error.html');
HTMLout = htmlrep(out, templatefile);
end
```

Podle uvedeného kódu dojde ke zobrazení výstupního formuláře s chybovým hlášením, pokud tato funkce zachytí chybu v případě, že není vyplněná některá z položek vstupního formuláře. Na obr. 3.2 je příklad s vykreslením výstupního formuláře error.html.



Obr. 3.2: Výstupní formulář s chybovým hlášením. Zdroj: vlastní.

3.3 Výstupní formulář

Každá MWS aplikace obsahuje výstupní formulář, což je klasický HTML soubor, který je uživatelům zobrazen v internetovém prohlížeči. Ve výstupním formuláři jsou výsledky

výpočtů či simulací (v podobě obrázků ve formátech bmp, jpeg apod.). Výstupní formulář je vygenerován vždy na konci m-souboru pomocí funkce *htmlrep* jako výsledek MWS aplikace.

Šablona výstupního formuláře se od běžné HTML stránky tím, že výstupní proměnné, které chceme vypsat, jsou vloženy mezi znaky $ $ s použitím atributu *autogenerate*. Pokud chceme, aby se matice zobrazila jako tabulka, je využit následující kód [14]:

```
<table autogenerate="$jmeno_matice$">
<tr>
<td>
</td>
</tr>
</table>
```

V případě, že je potřeba využít při výpočtech či simulacích Simulink, výstup je většinou v podobě obrázku. Nejprve využijeme proměnnou *mlid*, která obsahuje jedinečný řetězec, který se může používat při tvoření názvů výstupních obrázků, grafů, datových souborů apod. pomocí *mlid* můžeme tedy jednoduše rozlišit, kterému procesu vygenerovaná data patří. [14]

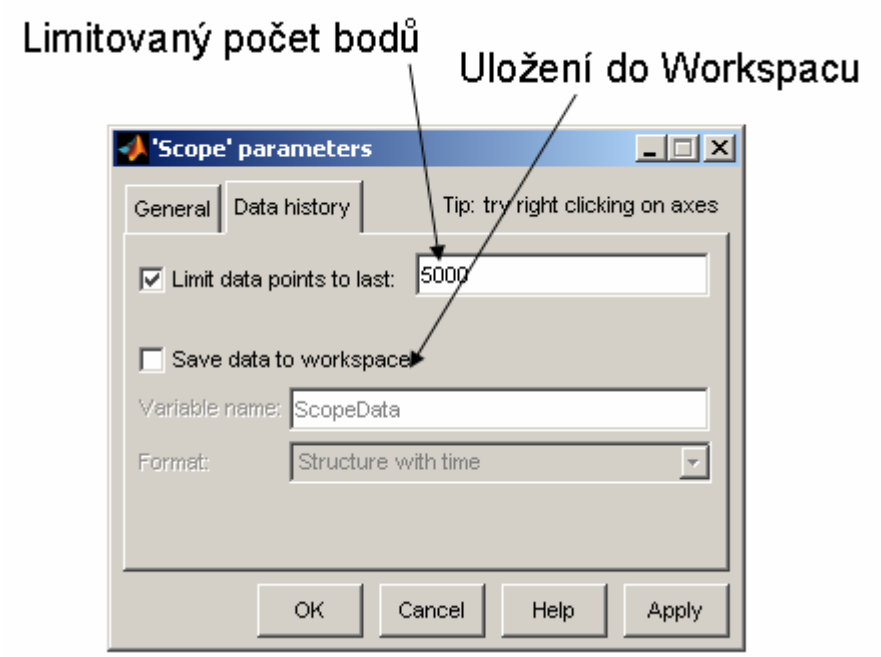
```
mlid = instruct.mlid;
```

V případě častých simulací dochází k vytvářením grafických výstupů se na pevný disk často ukládají grafické soubory. Tyto soubory je ale možné pravidelně vymazat využitím funkce *wscleanup*.

```
wscleanup('*.jpeg',0.2);
```

Nejdůležitější částí těchto činností je vytvoření grafických výstupů ze Simulinku. Prvním krokem je uložení dat ze Simulinku do pracovního prostoru MATLABu. Zde jsou využity vlastnosti bloku Scope, sloužící k zobrazení průběhů. V parametrech bloku Scope v položce *Data history* je nutné zaškrtnout pole *Save data to workspace*. Využitím této vlastnosti se simulovaná data uloží do pracovního prostoru. Následně pomocí funkce *Plot* se z uložených dat vytvoří graf. [13]

```
plot(ScopeData.time, ScopeData.signals.values);
```



Obr. 3.3: Vlastnosti bloku Scope. Zdroj: vlastní.

Pro zobrazení grafu ve výstupním formuláři se graf musí vložit v grafickém formátu. K tomuto se využívána funkce *wsprintjpeg*. Potom se do proměnné *oustruct* uloží vygenerovaný obrázek a prostřednictvím funkce *htmlrep* se zobrazí ve výstupním formuláři.

```
wsprintjpeg(f, s.GraphFileName);
s.GraphFileName = sprintf('/icons/regulator%s.jpeg', mlid);
```

Grafické soubory jsou zobrazeny využitím atributu *autogenerate*, které jsou vloženy mezi znaky $ $.

```
<img border=0 src="$GraphFileName$">
```

4 Příklady řešených úloh

Pro praktické využití MWS aplikace byly vytvořeny 4 příklady, které demonstrují využití jednak numerických výpočtů a také simulací. Veškeré zdrojové kódy MWS aplikace a také simulační schémata jsou na přiloženém CD-ROM. V MWS aplikaci jsou tyto řešeny tyto příklady:

- jednoduché regulátory,
- model hospodářské stability,
- řízení s korekčním členem
- stabilita systému.

4.1 Jednoduché regulátory

Regulátor je zařízení, provádějící regulaci, čili které prostřednictvím akční veličiny působí na regulovanou veličinu tak, aby se regulovaná veličina udržovala na předepsané hodnotě (ve zvláštních případech to nemusí být konstantní hodnota) a regulační odchylka byla nulová nebo co nejmenší. [12]

Podle průběhu výstupního signálu jsou regulátory děleny na spojité a nespojité. Spojité regulátory pracují se spojitými signály. Jejich hlavními „stavebními kameny“ jsou operační zesilovače. Nespojité regulátory pracují s nespojitými signály. Výstupem diskrétních signálů je posloupnost numerických hodnot. Jedná se zejména o číslicové počítače ve funkci regulátorů. Regulátory se také mohou dělit na lineární a nelineární. Zde je rozhodujícím prvkem statická charakteristika. V našem případě budeme pracovat s regulátory lineárními. [12]

Jednoduché regulátory se dělí na tři skupiny:

- proporcionální,
- integrační,
- derivační.

4.1.1 Jednotlivé typy regulátorů

Nejjednodušším příkladem regulátor je pouhé zesilování. V tomto případě je regulátor pouze zesilovač a akční veličina úměrná regulační odchylce.

$$u = r_0 e \tag{4.1}$$

Takovýto regulátor se nazývá proporcionální regulátor neboli P regulátor.Tento regulátor využívá ke své činnosti principu přímé úměrnosti – čím větší je detekovaná odchylka, tím

větší je i akční zásah. V praxi jsou tyto regulátory využívány zejména díky své jednoduchosti, dostatečně rychlému průběhu regulace stability Jejich největší nevýhodou je existence tzv. trvalé regulační odchylky, tj. nejsou schopny zcela odstranit rozdíl mezi skutečnou a žádanou hodnotou regulované veličiny. [12]

Druhým případem regulátoru může být takový regulátor, kdy je akční veličina úměrná integrálu regulační odchylky.

$$u = r_1 \int e dt \tag{4.2}$$

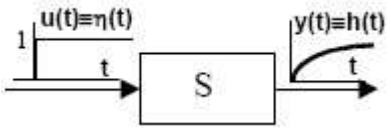
Jde o integrační regulátor neboli I regulátor. Je komplikovanějším regulátorem než-li proporcionální. Zde je změna akční veličiny úměrná časové hodnotě integrálu z regulační odchylky. Znamená to, že momentální rychlost změny akční veličiny závisí přímo na velikosti regulační odchylky. Tento typ regulátor je schopen zcela eliminovat regulační odchylku, tj. vrátit „vychýlenou“ hodnotu zpět na její požadovanou hodnotu. Na druhou stranu je ale tento typ méně stabilní než ostatní dva základní typy. [12]

Třetím a posledním typem je regulátor, kdy je akční veličina úměrná derivaci regulační odchylky.

$$u = r_1 e' \tag{4.3}$$

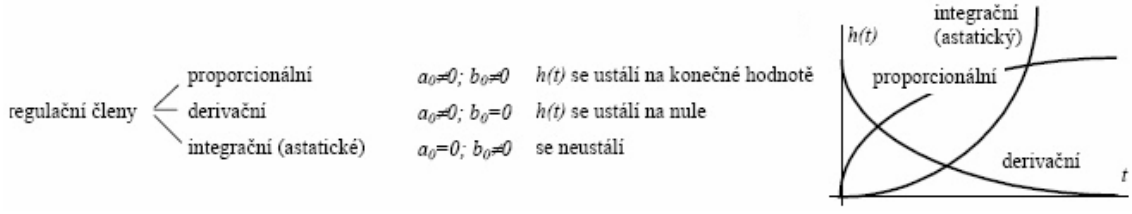
Je to případ regulátoru derivačního, neboli D regulátoru. Technická realizace, na rozdíl od dvou předchozích, není možná. V praxi se tento typ využívá hlavně v kombinaci s P regulátorem, nebo s PI regulátory. [12]

Přechodová funkce je odezvou na jednotkový skok a označujeme ji $h(t)$. Jejím grafickým záznamem je přechodová charakteristika. [6]



Obr. 4.1: Způsob sestrojení přechodové charakteristiky. Zdroj: [12],s .41.

Podle jednotlivých charakteristik základních typů regulátorů je vhodné určit základní rozdělení přechodových charakteristik na základě jejich průběhů a doby, kdy se ustálí.



Obr. 4.2: Charakteristiky regulačních členů. Zdroj: [12], s. 44.

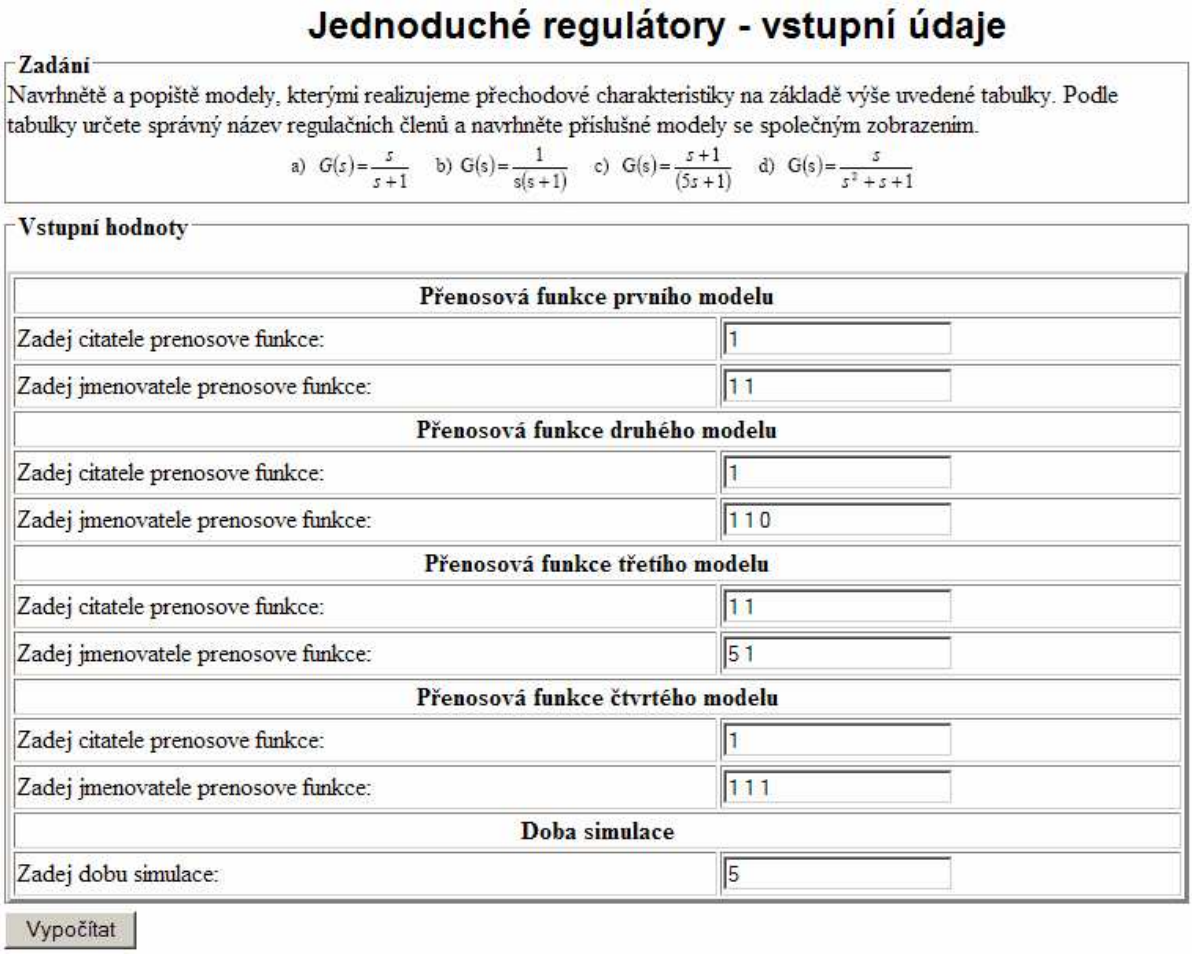
V tabulce 4.1 jsou uvedeny jednotlivé regulační členy s jejich odpovídajícími přenosy.[12]

Tab. 4.1: Přenosy regulačních členů. Zdroj: [12], s. 44.

člen	ideální	se zpožděním 1.řádu	se zpožděním 2.řádu	obecný
proporcionální	$G(s)=k$	$G(s)=\frac{k}{Ts+1}$	$G(s)=\frac{k}{(T_1s+1)(T_2s+1)}$	$G(s)=\frac{b_ms^m+\ldots+b_1s+b_0}{a_ns^n+\ldots+a_1s+a_0}$
derivační	$G(s)=ks$	$G(s)=\frac{ks}{Ts+1}$	$G(s)=\frac{ks}{(T_1s+1)(T_2s+1)}$	$G(s)=\frac{s(b_ms^m+\ldots+b_1s+b_0)}{a_ns^n+\ldots+a_1s+a_0}$
integrační	$G(s)=\frac{k}{s}$	$G(s)=\frac{k}{s(Ts+1)}$	$G(s)=\frac{k}{s(T_1s+1)(T_2s+1)}$	$G(s)=\frac{b_ms^m+\ldots+b_1s+b_0}{s(a_ns^n+\ldots+a_1s+a_0)}$

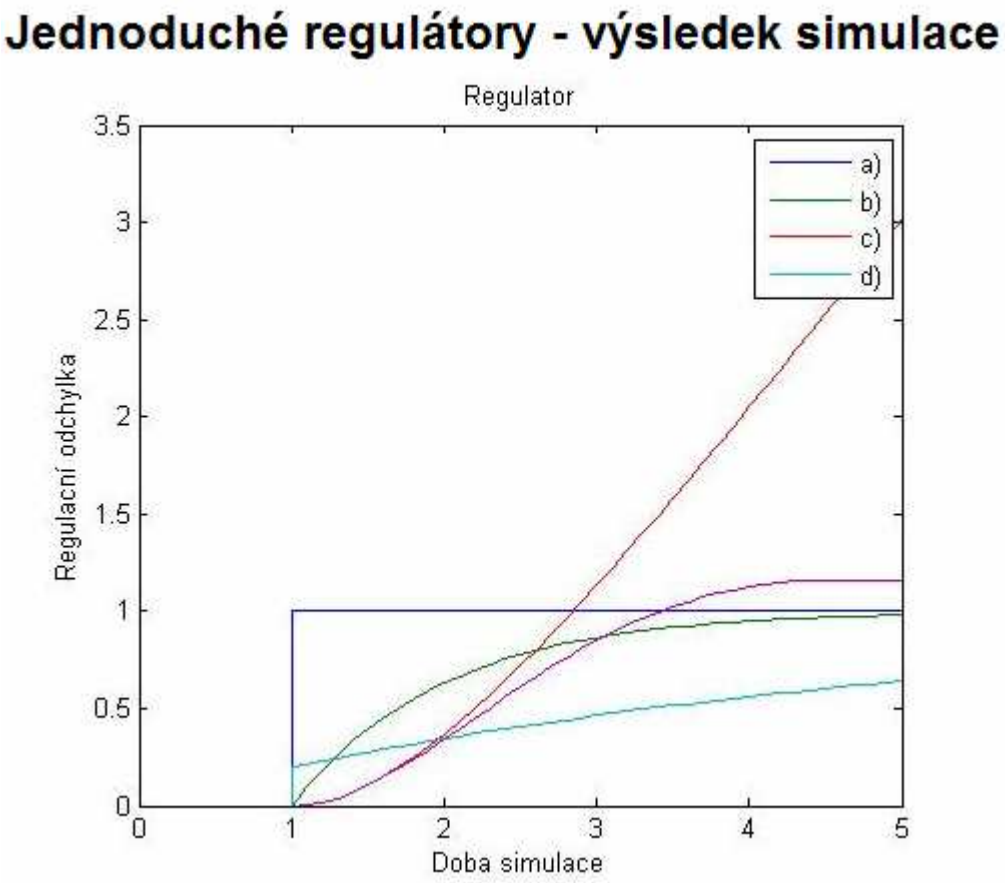
4.1.2 Příklad v MWS aplikaci

Jak bylo uvedeno v kapitole 3.1, pro MWS aplikaci je nejprve potřeba vytvořit vstupní formulář. Ve vstupním formuláři jsou obsaženy položky pro vložení vstupních údajů, např. přenosové funkce jednotlivých prvků řídícího obvodu a doba simulace pro tyto prvky. Příklad takovéhoto vstupního formuláře je na obr. 4.3.



Obr. 4.3: Vstupní formulář pro modely jednoduchých regulátorů. Zdroj: vlastní.

Po odeslání údajů stisknutím tlačítka „Vypočítat“ se údaje přenesou na HTTP server. Mezi údaji formuláře se pomocí skrytého pole *mlmfile* přenese i jméno funkce, které má zpracovávat zadané údaje. MWS potom spustí příslušnou funkci, vytvářející ze vstupních dat údaje, a objeví se ve výstupním formuláři. Pomocí funkce *htmlrep*, v níže je vstupním parametrem název výstupního formuláře a struktura s vypočítanými údaji. Po ukončení simulace jednoduchých regulátorů se uživateli vykreslí jejich přenosové charakteristiky ve výstupním formuláři.



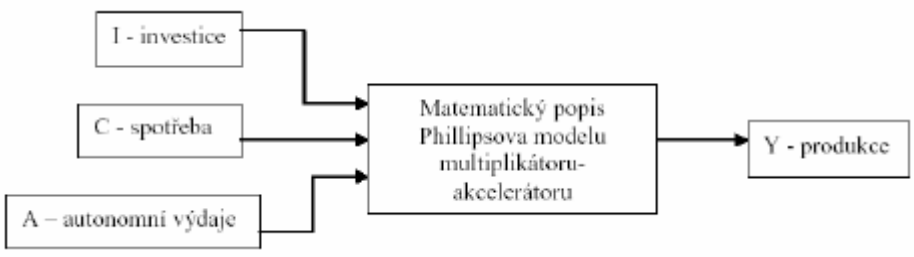
Obr. 4.4: Výstupní formulář pro modely jednoduchých regulátorů. Zdroj:vlastní.

4.2 Model hospodářské regulace

Model hospodářské regulace vychází z modelu multiplikátoru-akcelerátoru, kde je zavedeno zpoždění. Pokud má být použito spojitého postupu, vede k diferenciálním rovnicím, je vhodné zpoždění předpokládat spojitě. Tento model poprvé vypracoval A. W. H. Phillips v roce 1954. [1] Tento model zkoumá výkyvy a průběhy hospodářského cyklu a snaží se jim zabránit např. kompenzací poklesu poptávky nebo zmenšením amplitudy produkce.

4.2.1 Matematický popis modelu multiplikátoru-akcelerátoru

Předpokladem modelu je, že poptávka Z a produkce Y jsou sledovány odděleně, přičemž jedna se zpožďuje za druhou. Autonomní výdaje A na investice a spotřební zboží jsou považovány za pevně dané. Mohou být konstantní, či se předpokládá, že se mění určitým způsobem v čase. [1]



Obr. 4.5: Matematický model multiplikátoru-akcelerátoru. Zdroj: vlastní.

Matematický model Phillipsova multiplikátoru – akcelerátoru je na obr. 4.5. Vstupem do modelu jsou investice, spotřeba a autonomní výdaje (= konst.) a jeho výstupem je objem produkce daný tímto modelem. Investice závisí na koeficientu υ a rychlostí reakce χ. Jsou-li pak skutečné vyvolané investice v čase t podmíněni změnami produkce, jsou dány vztahem [1]:

$$\frac{dI}{dt} = -\chi\left(I - \upsilon\frac{dY}{dt}\right) \tag{4.4}$$

V tomto modelu celková poptávka je dána vztahem: $Z = C + I + A$, kde $C = cY = (1-s)Y$ bez zpoždění, z čehož vyplívá, že[1]:

$$Z = (1-s)Y + I + A \tag{4.5}$$

Na straně nabídky předpokládáme zpoždění o rychlosti reakce λ, takže platí:

$$\frac{dY}{dt} = -\lambda(Y - Z) \tag{4.6}$$

Vztahy popisující model multiplikátoru – akcelerátoru jsou (4.4), (4.5) a (4.6). Jsou zde přítomná dvě zpoždění, jedno na straně nabídky (produkce reaguje na poptávku s rychlostí λ) a druhé na straně akcelerátoru (vyvolané investice reagují na změny v produkci s rychlostí χ). Diferenciální rovnici pro Y získáme, pokud do rovnice (4.6) za Z dosadíme (4.5). Úpravou získáme diferenciální rovnici [1]:

$$\frac{d^2Y}{dt^2} + a\frac{dY}{dt} + bY = \chi\lambda A \tag{4.7}$$

$$D^2Y + aDY + bY = \lambda(D + \chi)A \tag{4.8}$$

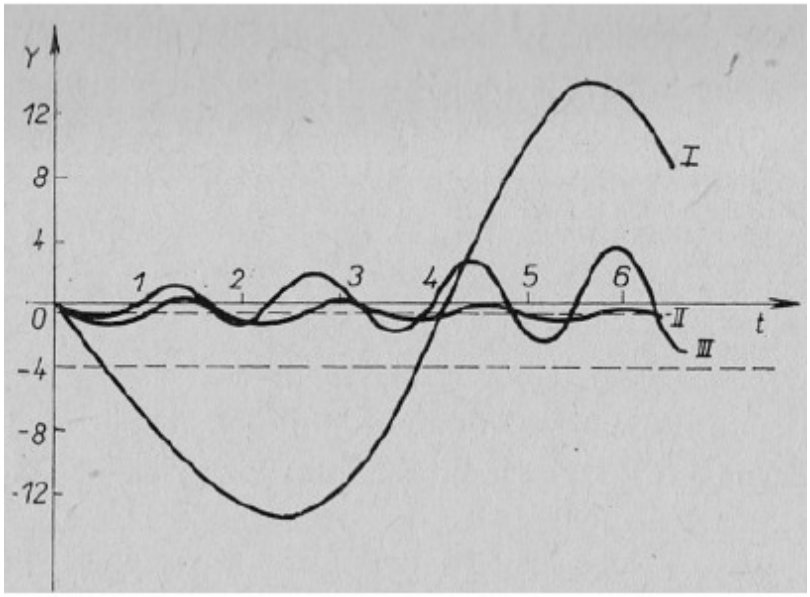
kde $a = \lambda s + \chi(1 - \lambda\upsilon)$, $b = \lambda\chi s$.

4.2.2 Stabilizační politika

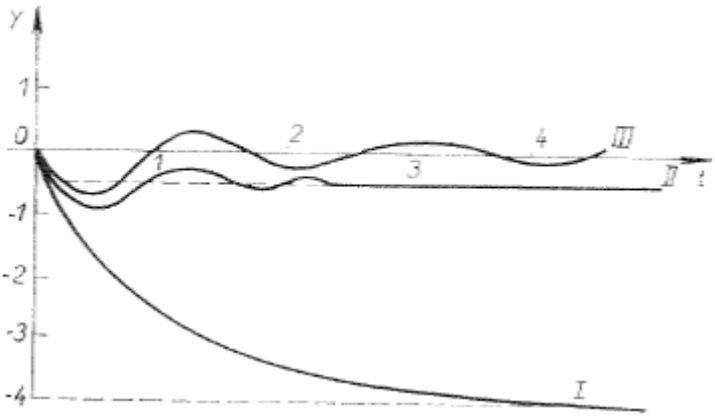
Podle Phillipse se rozlišují následující typy [1]:

1. *Proporcionální stabilizační politika*: státní poptávka je $\overline{G} = -f_p Y$, takže klesá-li Y pod požadovanou úroveň (Y=0), je poptávka tvořena vládou úměrná velikosti poklesu produkce pod tuto úroveň. Koeficient f_p znamená sílu stabilizační politiky.
2. *Integrální stabilizační politika*: státní poptávka je $\overline{G} = -f_i \int_0^t Y d\tau$, takže dodatečná poptávka vlády je úměrná kumulovanému poklesu produkce Y proti požadované úrovni za dobu od *0* do *t*.
3. *Derivační stabilizační politika*: státní poptávka $\overline{G} = -f_d DY$, takže vládní poptávka se řídí ne poklesem funkce produkce Y, ale rychlostí tohoto poklesu *(-DY)*

$\overline{G}$ je zde plánovaná poptávka vlády.



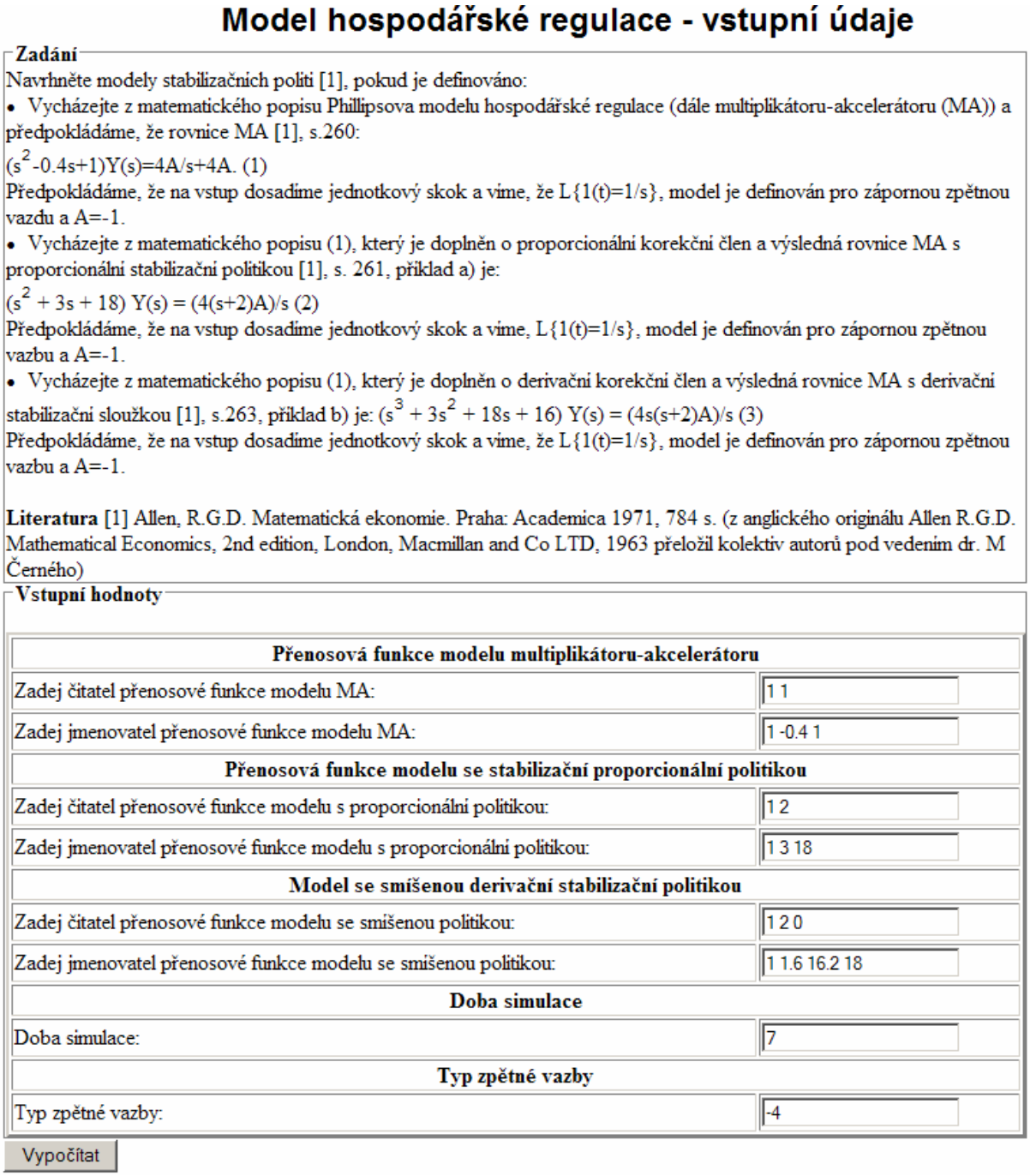
Obr. 4.6: Přechodové charakteristiky modelů. Zdroj: [1], s. 253.



Obr. 4.7: Přechodové charakteristiky modelů. Zdroj: [1], s. 253.

4.2.3 Příklad v MWS aplikaci

I v tomto případě je prvním kontaktem s uživatelem vstupní formulář, obsahující položky pro ukládání vstupních údajů, např. přenosové funkce jednotlivých typů politik, doba simulace a typ zpětné vazby. Příklad tohoto vstupního formuláře, kde si uživatel může simulovat účinky jednotlivých typů stabilizačních politik, je na obr. 4.8



Model hospodářské regulace - vstupní údaje

Zadání

Navrhněte modely stabilizačních politi [1], pokud je definováno:

- Vycházejte z matematického popisu Phillipsova modelu hospodářské regulace (dále multiplikátoru-akcelerátoru (MA)) a předpokládáme, že rovnice MA [1], s.260:

$(s^2-0.4s+1)Y(s)=4A/s+4A$. (1)

Předpokládáme, že na vstup dosadíme jednotkový skok a víme, že L{1(t)=1/s}, model je definován pro zápornou zpětnou vazdu a A=-1.

- Vycházejte z matematického popisu (1), který je doplněn o proporcionální korekční člen a výsledná rovnice MA s proporcionální stabilizační politikou [1], s. 261, příklad a) je:

$(s^2 + 3s + 18)\ Y(s) = (4(s+2)A)/s$ (2)

Předpokládáme, že na vstup dosadíme jednotkový skok a víme, L{1(t)=1/s}, model je definován pro zápornou zpětnou vazbu a A=-1.

- Vycházejte z matematického popisu (1), který je doplněn o derivační korekční člen a výsledná rovnice MA s derivační stabilizační sloužkou [1], s.263, příklad b) je: $(s^3 + 3s^2 + 18s + 16)\ Y(s) = (4s(s+2)A)/s$ (3)

Předpokládáme, že na vstup dosadíme jednotkový skok a víme, že L{1(t)=1/s}, model je definován pro zápornou zpětnou vazbu a A=-1.

Literatura [1] Allen, R.G.D. Matematická ekonomie. Praha: Academica 1971, 784 s. (z anglického originálu Allen R.G.D. Mathematical Economics, 2nd edition, London, Macmillan and Co LTD, 1963 přeložil kolektiv autorů pod vedením dr. M Černého)

Vstupní hodnoty

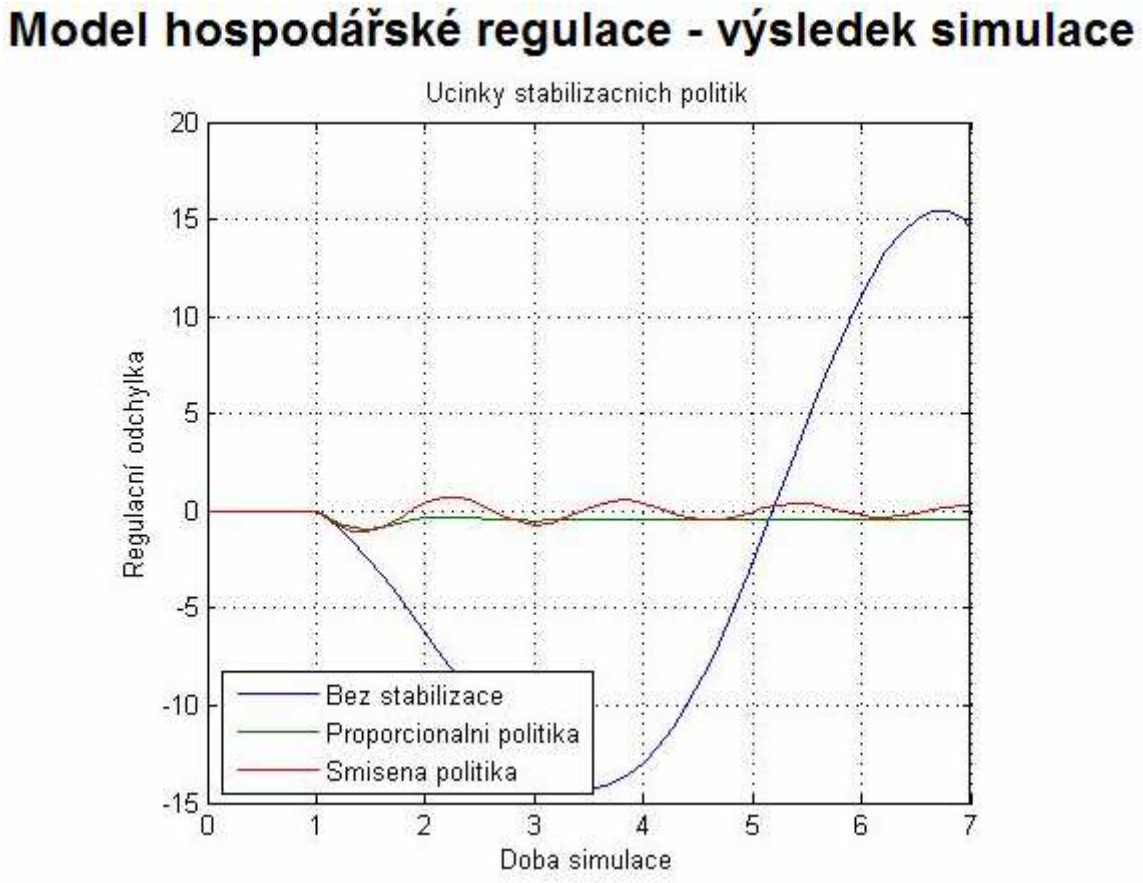
Přenosová funkce modelu multiplikátoru-akcelerátoru	
Zadej čitatel přenosové funkce modelu MA:	1 1
Zadej jmenovatel přenosové funkce modelu MA:	1 -0.4 1
Přenosová funkce modelu se stabilizační proporcionální politikou	
Zadej čitatel přenosové funkce modelu s proporcionální politikou:	1 2
Zadej jmenovatel přenosové funkce modelu s proporcionální politikou:	1 3 18
Model se smíšenou derivační stabilizační politikou	
Zadej čitatel přenosové funkce modelu se smíšenou politikou:	1 2 0
Zadej jmenovatel přenosové funkce modelu se smíšenou politikou:	1 1.6 16.2 18
Doba simulace	
Doba simulace:	7
Typ zpětné vazby	
Typ zpětné vazby:	-4

Vypočítat

Obr. 4.8: Vstupní formulář pro modely stabilizačních politik. Zdroj: vlastní.

Po odeslání údajů stisknutím tlačítka „Vypočítat“ se údaje přenesou na HTTP server. Mezi skrytými údaji se přenese i jméno funkce pomocí skrytého pole *mlmfile*, který zpracovává údaje. MWS potom spustí příslušnou funkci vytvářející ze vstupních dat údaje,

které se objeví ve výstupním formuláři. Využitím funkce *htmlrep*, kde je vstupním parametrem název výstupního formuláře a struktura s vypočítanými údaji. Po ukončení simulace se uživateli vykreslí jejich přenosové charakteristiky ve výstupním formuláři.



Obr. 4.9: Výstupní formulář pro modely hospodářské regulace. Zdroj: vlastní.

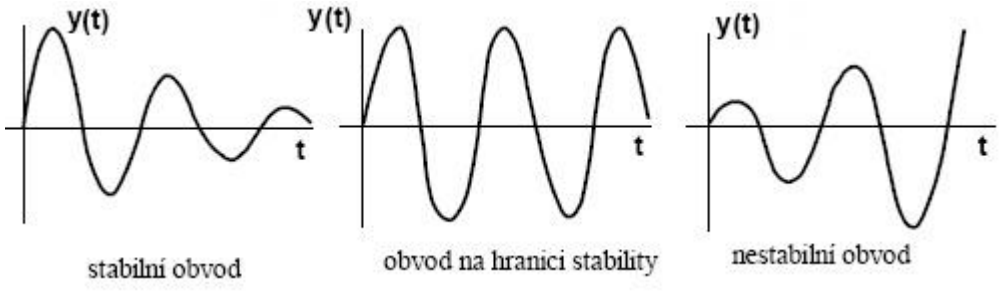
4.3 Stabilita regulačních obvodů

Stabilita je základní a nevyhnutelnou podmínkou správné funkce regulačního obvodu. Regulační obvodu je možné považovat za stabilní, jestliže se po svém vychýlení z rovnovážného stavu a odstranění vzruchu, který vychýlení způsobil, je schopen se ustálit v rovnovážném stavu. Nový rovnovážný stav ale nemusí být stejný, jako rovnovážný stav předchozí. [12]

4.3.1 Stabilita spojitých systémů

Stabilitu můžeme považovat za schopnost regulačního obvodu, kdy se jeho regulovaná veličina *y* (respektive její přechodová složka *y(t)*) ustálí na původní hodnotě po vychýlení poruchovou veličinou nebo při vychýlení řídící veličinou na nové hodnotě.

Z hlediska stability se regulační obvody dělí na: stabilní, na hranici stability a nestabilní. Mezní stav, kdy *y(t)* kmitá kmity o stabilní amplitudě, je nazýván hranicí stability. [12]



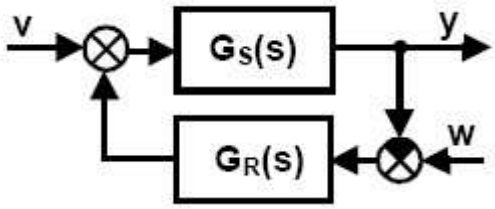
Obr. 4.10: Stabilita regulačního obvodu. Zdroj: [12], s. 67.

Stabilitu regulačního obvodu lze matematicky vyjádřit takto:

$$\lim_{t\to 0} y(t) = 0 \tag{4.9}$$

Objevuje se problém, zda poznáme, kdy je navrhovaný regulační obvod stabilní či nestabilní. Nejprve si objasníme obecnou podmínku stability.

Mějme regulační obvod podle obr. 4.11. Jeho přenos řízení a přenos poruchy je dán rovnicemi (4.10) a (4.11), které si zavedeme jako podíl obecných polynomů.



Obr. 4.11: Jednoduchý regulační obvod. Zdroj: [12], s. 67.

$$G_W(s) = \frac{Y(s)}{W(s)} = \frac{G_0(s)}{1+G_0(s)} = \frac{b_m s^m + ... + b_1 s + b_0}{a_n s^n + ... + a_1 s + a_0} \tag{4.10}$$

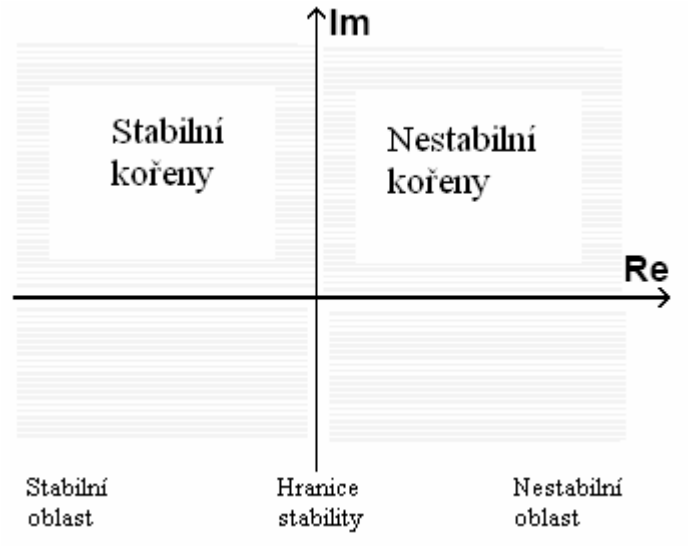
$$G_V(s) = \frac{Y(s)}{V(s)} = \frac{G_s(s)}{1+G_0(s)} = \frac{c_m s^m + ... + c_1 s + c_0}{a_n s^n + ... + a_1 s + a_0} \tag{4.11}$$

Pokud položíme jmenovatel přenosu řízení nebo poruchy (jsou vždy stejné) roven nule, získáme charakteristickou rovnici regulačního obvodu.

$$1 + G_0(s) = 0 \tag{4.12}$$

$$a_n s^n + ... + a_1 s + a_0 = 0 \tag{4.13}$$

Regulační obvod je stabilní, pokud jsou všechny kořeny $s_1, s_2, ... s_n$ charakteristické rovnice (4.12) respektive (4.13) záporná čísla a v případě komplexních kořenů mají tyto kořeny zápornou reálnou část. Obecně je regulační obvod považován za stabilní, pokud jsou všechny kořeny charakteristické rovnice v záporné reálné části neboli leží v levé komplexní polorovině.



Obr. 4.12: Poloha kořenů charakteristické rovnice. Zdroj:[12], s. 68.

V případě, že některý z kořenů charakteristické rovnice leží na imaginární ose a žádný neleží v pravé komplexní polorovině, je obvod na hranici stability.

Rovnice (4.12) v podstatě není charakteristickou rovnicí, ale úpravou lze snadno získat. Postup při sestavování charakteristické rovnice je následující. Přenos regulačního obvodu, který je součinem přenosu soustavy a přenosu regulátoru a je vyjádřen ve tvaru podílu polynomů.

$$G_{0(s)} = G_R(s)G_s(s) = \frac{M(s)}{N(s)} \tag{4.14}$$

Potom můžeme napsat

$$1 + G_{0(s)} = 1 + \frac{M(s)}{N(s)} = \frac{M(s) + N(s)}{N(s)} = 0 \tag{4.15}$$

a protože zlomek je roven nule, když jeho čitatel je roven nule, můžeme charakteristickou rovnici psát jako součet polynomů čitatele a jmenovatele regulačního obvodu $G_0(s)$.

$$M(s) + N(s) = 0 \tag{4.16}$$

Aby byl regulační obvod stabilní, musí být všechny koeficienty charakteristické rovnice (4.13) kladné. Toto je nutná, ale nikoli postačující podmínka. [12]

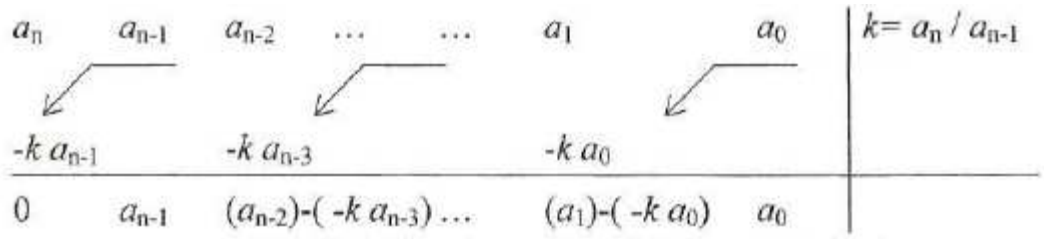
Existuje řada kritérií stability, které je možné rozdělit na dvě základní skupiny [6]:

- algebraická kritéria, které vycházejí z diferenciálních rovnice nebo přenosové funkce. K nejpoužívanějším kriteriím patří Routh-Schurovo kritérium či Hurwitzovo kritérium, využívající maticový počet;
- frekvenční kritéria, vycházející z frekvenční charakteristiky. Zde můžeme uvést jako příklad kritéria Nyquistovo.

Routh-Schurovo kritérium vychází z charakteristické rovnice (4.12) regulačního obvodu. Regulační obvod je stabilní, pokud jsou koeficienty charakteristické rovnice a všechny koeficienty „redukovaných“ rovnic až do druhého řádu. [6]

Algoritmus Routh-Schurova kritéria je následující [6,12]:

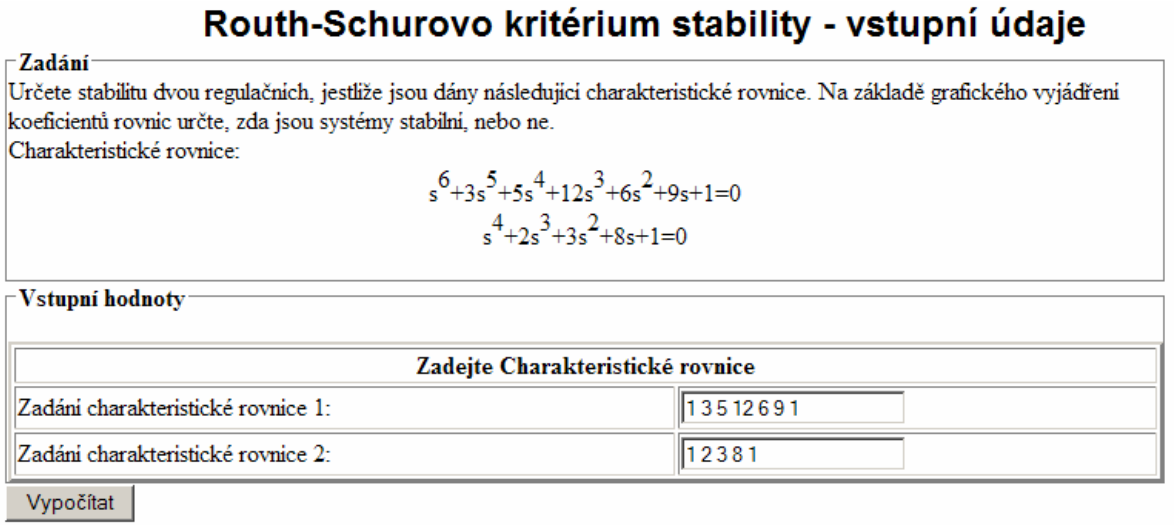
- vypíšeme koeficienty charakteristického do řádku od nejvyšší mocniny k nejnižší;
- každý druhý koeficient (zleva) podtrhneme;
- určíme redukční konstantu, která je podílem dvou nejvyšších koeficientů $k = a_n / a_{n-1}$;
- podtržené koeficienty vynásobíme konstantou k a výsledek napíšeme o 1 místo doleva na následující řádek;
- řádky odečítáme a dostaneme koeficienty charakteristické rovnice o 1 řád nižší;
- 2 až 5 bod se opakuje tak dlouho, až dostaneme rovnici druhého řádu (je představována koeficienty $(a_2, a_1 a_0)$. Pokud je během redukce jeden z koeficientů záporný, není nutné dále pokračovat, protože systém je nestabilní.



Obr. 4.13: Algoritmus Routh-Schurova kritéria. Zdroj: [6], s. 94.

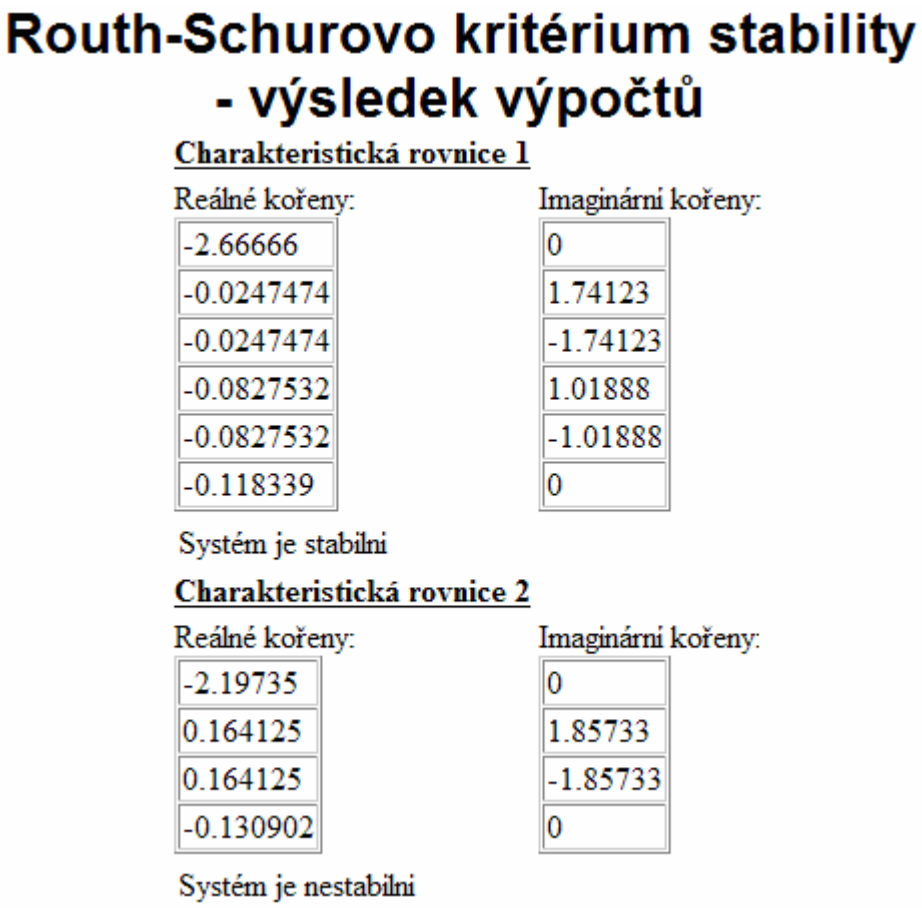
4.3.2 Příklad v MWS aplikaci

I zde je pro tuto simulaci nejprve nutné vytvořit vstupní formulář. Zde se zadávají charakteristické rovnice, pro které se vypočítají jejich kořeny a určí se stabilita systémů. Příklad tohoto vstupního formuláře, kde chce uživatel vypočítat stabilitu systémů, je na obrázku 4.14.



Obr. 4.14: Vstupní formulář pro výpočet stability systémů. Zdroj: vlastní.

Po odeslání údajů stisknutím tlačítka „Vypočítat“ se údaje přesunou na HTTP server. Současně s údaji z formuláře se také přenese pomocí *mlmfile* i jméno funkce, která má zadané údaje zpracovávat. MWS potom spustí příslušnou funkci, kde se vytvoří ze vstupních dat údaje, a jsou zobrazena ve výstupním formuláři. Využitím funkce *htmlrep*, kde jsou parametry název výstupního formuláře a struktura s vypočítanými údaji. Výsledkem této funkce je řetězec s výsledným kódem HTML stránky. Po ukončení výpočtů se uživateli zobrazí kořeny charakteristických rovnic a zda jsou systémy stabilní, nebo nejsou.



Obr. 4.15: Výstupní formulář s výpočty stability systémů. Zdroj: vlastní.

4.4 Řízení s korekčními členy

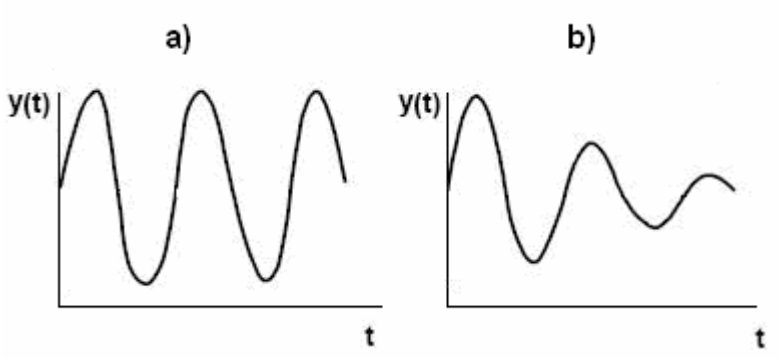
Při návrhu korekčního členu je potřeba si uvědomit, že je nutné korigovat ty vlastnosti řídícího obvodu, které způsobují nestabilitu obvodu. Jde o zařazení prvků do řídícího systému a případně ještě může jít o zavedení vnitřní zpětné vazby. Musíme navrhnout takový korekční člen, aby člen původního řídícího obvodu, způsobující nestabilitu byl eliminovaný, tj. aby se vzájemně vykrátili. Cílem je tedy ovlivnit amplitudu signálu a fázi. Korekce můžeme provést změnou zesílení nebo pomocí korekčního členu. [2]

4.4.1 Druhy korekcí

Existují tři základní druhy korekcí, které jsou v praxi využívány:

- korekce změnou zesílení;
- sériová korekce;
- zpětnovazební korekce.

Při korekci změnou zesílení můžeme zesílení snížit, což může mít stabilizující účinek na řídící obvod. V komplexní rovině dojde ke zmenšení amplitudy frekvenční charakteristiky.

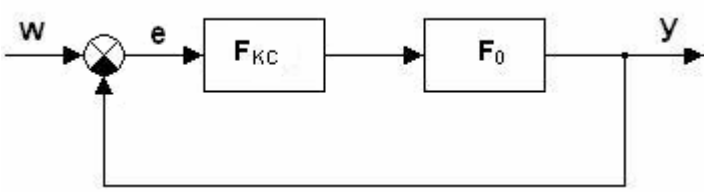


Obr. 4.16: Korekce změnou zesílení. Zdroj: vlastní.

Řídící systém je nestabilní, pokud jeho přechodová má tvar podle obr. 4.16, charakteristika a. Korekcí pomocí zesílení dojde k ustálení kmitů a systém se stabilizuje na určité hodnotě, což ukazuje přechodová charakteristika podle obr. 4.16 b.

Při sériové korekci existují tyto typy korekčních členů [2]:

- derivační korekční člen;
- integrační korekční člen;
- integro-derivační korekční člen.



Obr. 4.17: Zapojení sériového korekčního členu. Zdroj: vlastní

Pomocí sériového korekčního členu se potlačují ty vlastnosti, způsobující nestabilitu obvodu. Uvedeme příklad řídícího obvodu, popsaný přenosovou funkcí:

$$F_{OK}(p) = F_O(p) \cdot F_{KC}(p) \tag{4.17}$$

Přenosy členů řídícího obvodu se mezi sebou vynásobí. Při praktických návrzích řídících obvodů volíme:

$$F_O(p) = \frac{k}{p \cdot (T_1 \cdot p + 1) \cdot (T_2 \cdot p + 1)} \tag{4.18}$$

kde $F_0(p)$ je přenosová funkce otevřeného obvodu a $T1>T2$. První časová konstanta T_1 označuje nevyšší časovou konstantu v celém obvodu. Druhá časová konstanta T_2 je několikanásobně menší než-li časová konstanta T_1, obvykle se volí hodnota 10kát menší.

Přenosová funkce korekčního členu má tvar:

$$F_{kc}(p) = K_d \frac{T_d \cdot p + 1}{T_a \cdot p + 1} \tag{4.19}$$

kde $T_d > T_a$, $T_d = T_1$.

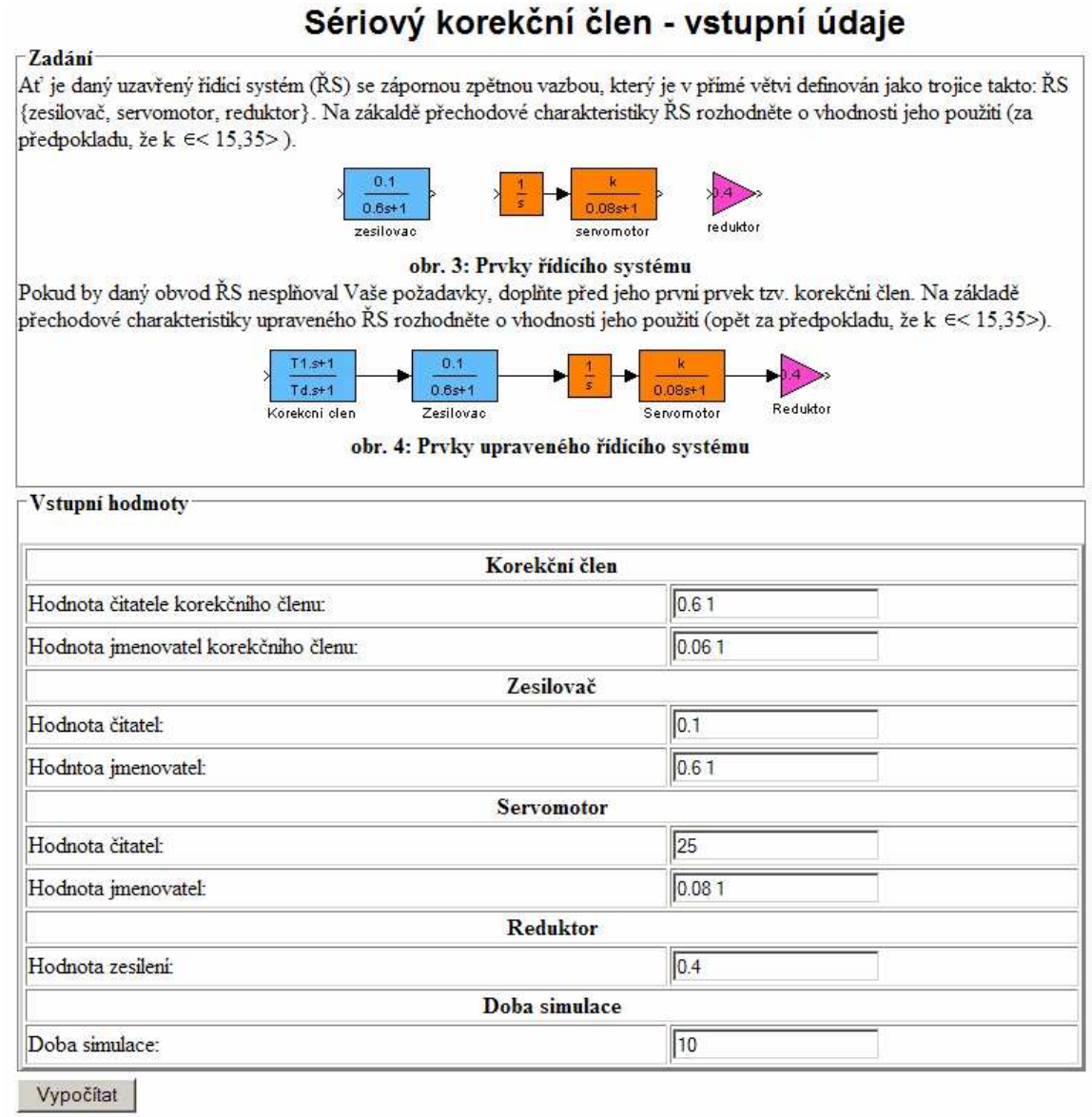
Přenosová funkce otevřeného řídícího obvodu je ve tvaru:

$$F_{OK}(p) = \frac{K \cdot K_d (T_d \cdot p + 1)}{p \cdot (T_1 \cdot p + 1)(T_2 \cdot p + 1)(T_a + 1)} = \frac{K_k}{p \cdot (T_2 \cdot p + 1)(T_a \cdot p + 1)} \tag{4.20}$$

.

4.4.2 Příklad v MWS aplikaci

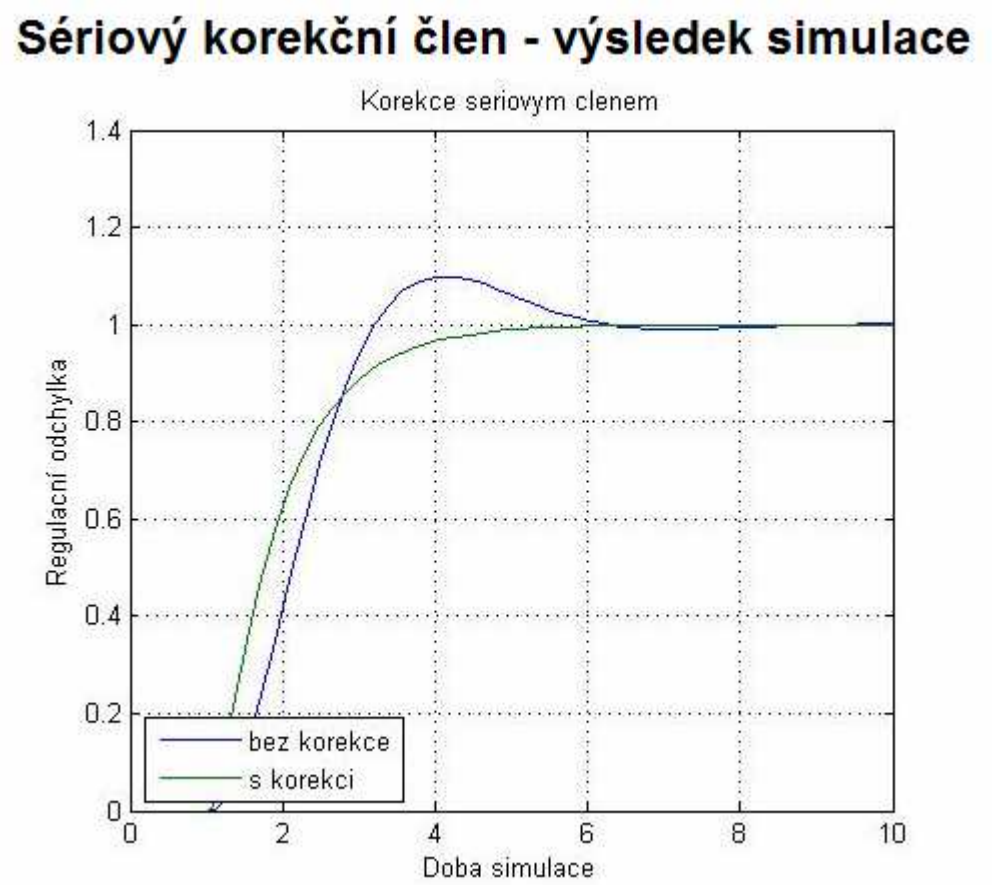
I zde je prvním krokem vytvoření vstupního formuláře, obsahující položky, sloužící pro ukládání vstupních údajů, např. přenosové funkce korekčního členu, servomotoru či hodnotu zesílení korektoru atd. příklad takovéhoto formuláře, kde si uživatel může simulovat účinnost korekčního členu, je na obr. 4.18.



Obr. 4.18: Vstupní formulář pro model sériového korekčního členu. Zdroj:vlastní.

Vložená data se odešlou stisknutím tlačítka „Vypočítat“ se přenesou na HTTP server. Mezi těmito údaji se přenese i jméno funkce pomocí skrytého pole *mlmfile*, pomocí níž

MWS spustí příslušný m-soubor. Spuštěný m-soubor vytvoří ze vstupních dat údaje, které se objeví ve výstupním formuláři využitím funkce *htmlrep*, do níž vstupuje jako parametr název souboru se šablonou výstupního formuláře a struktura s vypočítanými údaji. Po ukončení simulace se uživateli vykreslí přenosové charakteristiky modelu bez korekčního členu a modelu s korekčním členem.



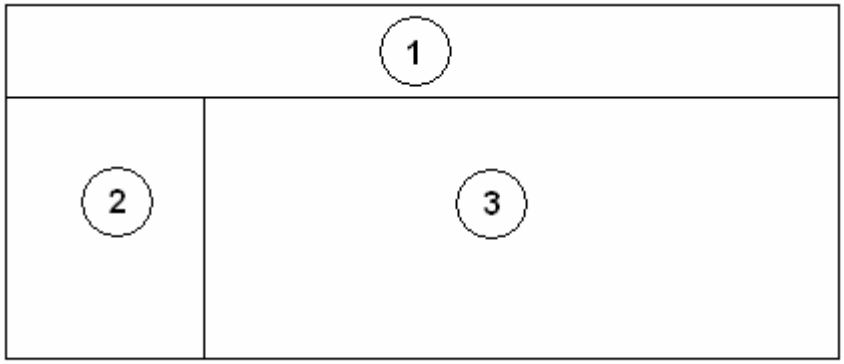
Obr. 4.19: Výstupní formulář pro model se sériovým korekčním členem. Zdroj: vlastní.

5 Struktura webových stránek MWS aplikace

Pro lepší orientaci může uživatelům sloužit mapa webových stránek, pomocí níž se uživatelé mohou lépe orientovat v MWS aplikaci. V této kapitole bude ukázána struktura webových stránek MWS aplikace, obsahující teoretický výklad společně s ukázkovými příklady, struktura webových stránek vstupních formulářů a nakonec mapa webových stránek MWS aplikace.

5.1 Struktura webových stránek MWS aplikace

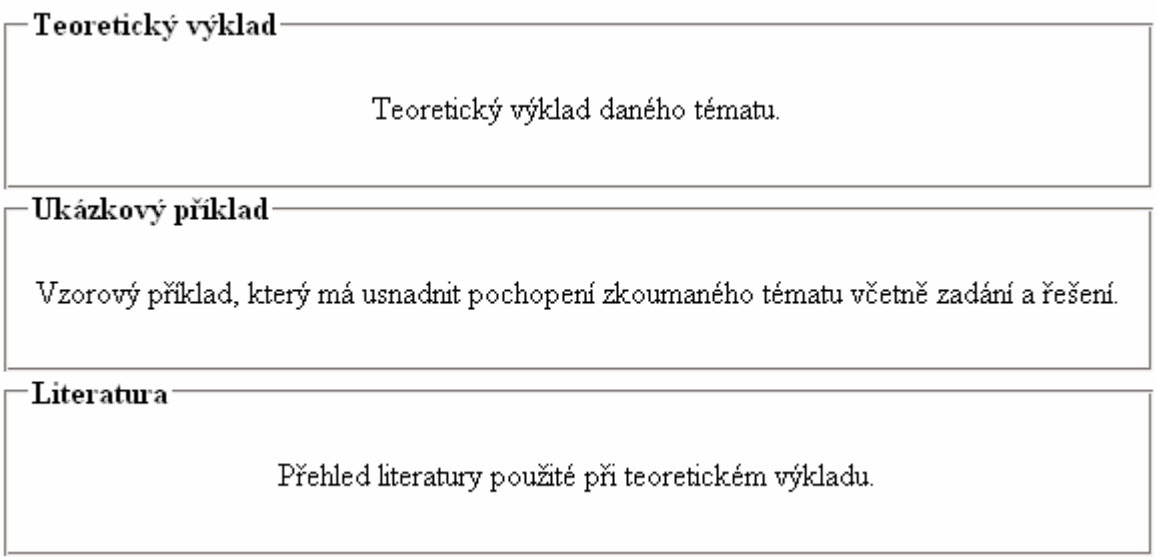
Webové stránky byly vytvořeny tak, aby orientace v nich byla co nejjednodušší. První oblastí je horní pruh (na obr. 5.1 oblast 1) v celé šířce plochy webové stránky. Obsahuje název každé úlohy či teoretického výkladu. Druhou oblastí je pruh, nacházející se na levé straně (na obr. 5.1 oblast 2). V této části se nachází přehledné navigační menu, sloužící k jednoduché orientaci. Poskytuje rychlé přepínání mezi jednotlivými tématy a stránkami MWS aplikace. Poslední oblast webové stránky slouží k teoretickému výkladu, zadávání vstupních údajů a zobrazování výsledků (na obr. 5.1 oblast 3).



Obr. 5.1: Rozdělení webové stránky MWS aplikace. Zdroj: vlastní.

Jak již bylo zmíněno, navigační prvky se podle obr. 5.1 nacházejí v levé oblasti webové stránky MWS aplikace. Toto umístění navigačního menu bylo zvoleno z toho důvodu, že se jedná o obvyklou pozici pro navigační prvky. Je to z toho důvodu, že uživatelé prohlížejí informace zleva doprava, proto se zaměří na navigaci jako první. [8]

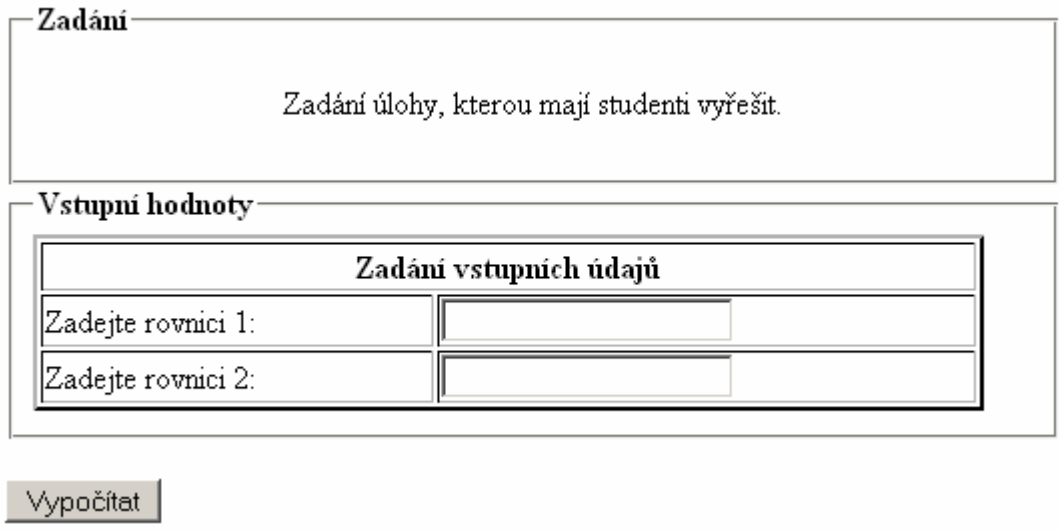
V hlavní části webové stránky je teoretický výklad k jednotlivým úlohám, a je rozdělen na tři základní bloky. V prvním bloku je teorie nezbytná k pochopení daného problému. Ve druhém bloku je ukázkový příklad, usnadňující uživateli pochopení daného problému. Nachází se zde zadání příkladu a jeho řešení. Ve třetím, posledním bloku, je seznam použité literatury, na kterou se v teoretickém výkladu odkazuje a umožňuje uživatelům, v případě zájmu, získat další informace.



Obr. 5.2: Uspořádání hlavní části webové stránky. Zdroj: vlastní.

5.2 Struktura vstupního formuláře MWS aplikace

Jednou z hlavních částí MWS aplikace je vstupní formulář, ve němž se zadávají vstupní údaje, dále zpracovávané MATLABem. Webová stránka se vstupním formulářem je rozdělena na dva základní bloky. V prvním bloku je zadání úlohy včetně potřebných údajů, zadávané později uživatelem do vstupního formuláře. Vstupní formulář je ve druhém bloku této stránky a je v podobě jednoduchého formuláře, umožňující přehledně zadávat vstupní údaje. Po zadání vstupních údajů dojde stisknutím tlačítka „Vypočítat“ k odeslání dat do MATLABu.

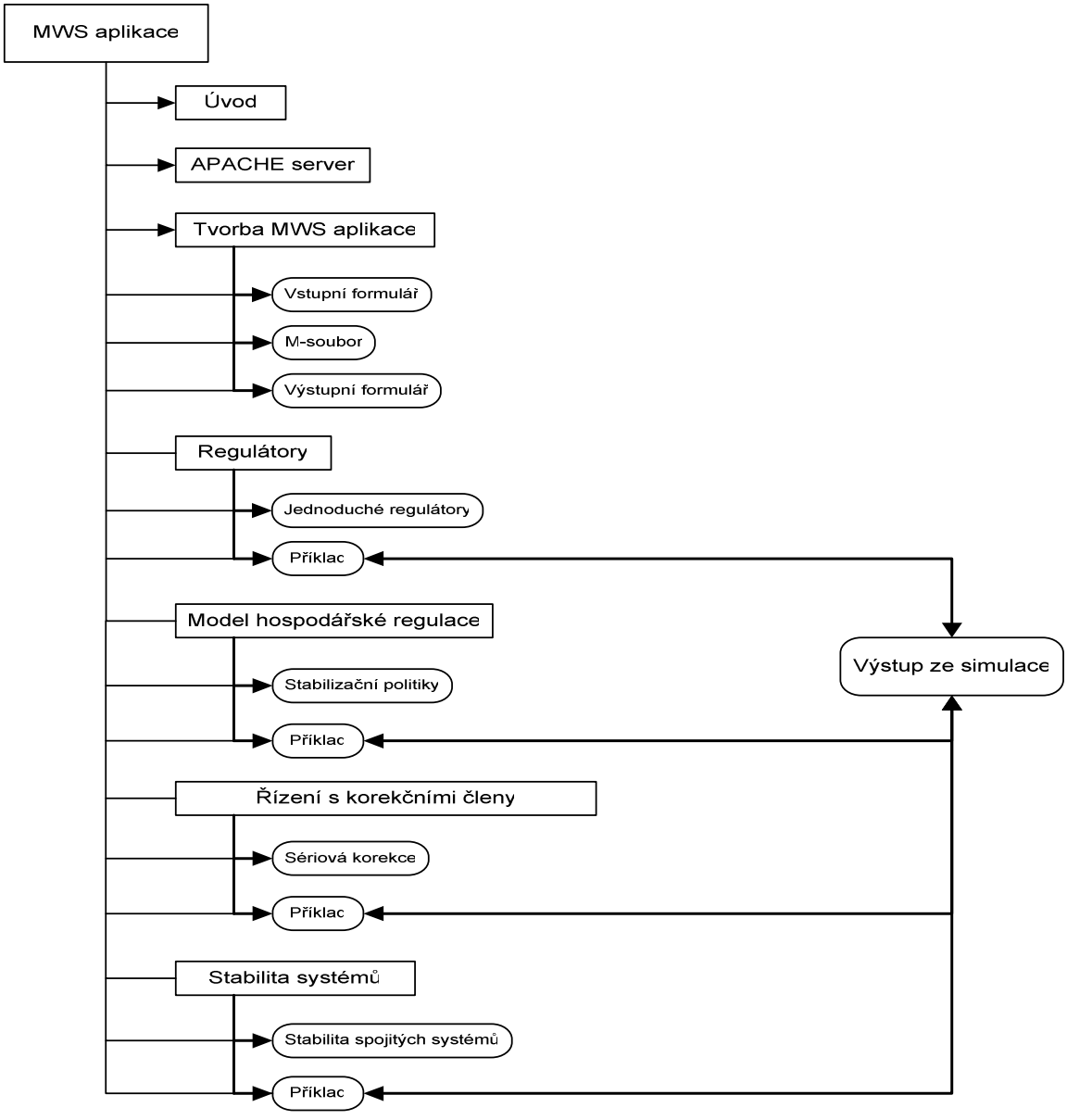


Obr. 5.3: Struktura vstupního formuláře. Zdroj: vlastní.

5.3 Mapa webových stránek MWS aplikace

Mapa graficky znázorňuje umístění různých prvků v určité oblasti nebo prostoru. V síti WWW je mapa webových stránek strukturálním přehledem webových stránek, která zobrazuje stránky ve vzájemném obecném vztahu podle strukturální blízkosti, případně, podle podobnosti témat. [8]

Každá stránka se věnuje určitému tématu, např. výklad o tvorbě MWS aplikací či již jednotlivé teoretické výklady v tématům z předmětu Teorie systémů.



Obr. 5.4: Mapa stránek MWS aplikace. Zdroj: vlastní.

6 Problémy při tvorbě MWS aplikace

Při tvorbě a provozu MWS aplikace je možné narazit na několik problémů. V této kapitole jsou uvedeny nejčastější problémy, překážky, které se při tvorbě MWS aplikace vyskytují nejčastěji .

Jedním z častých problémů je, že po vytvoření MWS aplikace, správného vyplnění vstupního formuláře po odeslání údajů internetový prohlížeč zobrazí následující chybové hlášení:

```
Error: Invalid configuration entry on server: priklad.m.
```

Jde o chybové hlášení MATLAB Web Serveru, že nenašel požadovaný m-soubor, v tomto případě priklad.m. Tento m-soubor je potřebný pro výpočet, simulaci a zobrazení výsledků ve výstupním formuláři. V takovém to případě je vhodné zkontrolovat konfigurační soubor matweb.conf, zda je v něm daná MWS aplikace zapsaná a jestli je správně zadaná adresa umístění m-souboru na pevném disku. Pokud je konfigurační soubor v pořádku, může se chyba vyskytnout v HTML kódu vstupního formuláře. Proto je třeba zkontrolovat parametr VALUE v tagu `<input>…</input>`, zda obsahuje správný název m-souboru.

Druhým problémem, který se při tvorbě MWS aplikace vyskytl, bylo zobrazování grafických výstupů ze SIMULNIKu ve výstupním formuláři. Řešením je využití vlastností bloku Scope, kde lze v nastavení naměřené hodnoty uložit do pracovního adresáře MATLABu. S těmito daty lze dále pracovat jako s klasickými daty (kap. 3.3).

Dalším problémem při tvorbě MWS aplikace je podpora pouze HTML kódu a nikoliv skriptovacích jazyků, např. PHP nebo JavaScript. Toto lze vyřešit např. pomocí rámů, kde si lze www stránku rozdělit na jednotlivé rámy podle potřeby. Rámy je možné využít právě v tomto, kdy MWS nepodporuje skriptovací jazyky a výstupní formulář je možné zobrazit bez jakýchkoliv dalších požadavků na úpravu funkčnosti či vzhledu. Po stisknutí tlačítka „Vypočítat“ ve vstupním formuláři se změní pouze ten rám, ve kterém se mění stránky podle požadavků uživatelů. Co se týče využívání externích kaskádových stylů, je potřeba při načítání uvést celou adresu umístění externího souboru, tedy uvést adresu absolutní, a ne pouze adresu relativní, jak je v běžných případech programování www stránek běžné. Tento problém, tedy že není podpora skriptovacích jazyků, je již vyřešen v nové verzi MATLABu podporou vytváření internetových aplikací pomocí JavaScriptu či .NET FRAMEWORKu.

Další překážkou pro tvůrce m-souborů může být formát dat, které MATLAB obdrží ze vstupního formuláře. Je nutné si uvědomit, že i když uživatel z jeho pohledu zadává čísla, v MATLABu jsou tyto data považována za textový řetězec. Pro převod takovéhoto řetězce je možné využít funkcí zmíněných v kap. 3.2.1. Je třeba si uvědomit, že pokud po výpočtech chce uživatel data zobrazit ve výstupním formuláři, musí tyto data opět převést zpět z čísel na řetězec, aby došlo k jejich správnému zobrazení.

Posledním problémem je vytvoření grafiky ve výstupních formulářích. Vstupní data se nejprve musí převést pomocí funkce str2num. Po provedení simulace si lze nasimulovaná data uložit do pracovního prostoru a již s nimi pracovat jako s klasickým daty. K vytvoření grafu se využije funkce Plot. Je také možné určit si velikost grafu pomocí funkce set.

Závěr

Cílem této práce je tvorba internetové učebnice, jenž má sloužit potřebám výuky pro předmět Teorie systémů. Pro tvorbu výpočetních a simulačních úloh byl využit výpočetní systém MATLAB a simulační nástroj Simulink. Součástí MATLABu je i toolbox MATLAB Web Server, umožňující propojení klasických WWW stránek a MATLABu. Samotnou práci lze rozdělit na dvě základní části. První část se zabývá popisem základních pojmů a procesem tvorby MWS aplikace. Druhá část se zabývá popisem úloh vytvořených v MWS aplikaci.

Při tvorbě MWS aplikace došlo propojením jednotlivých částí, tj. WWW stránek a MATLAB Web Serveru, k vytvoření internetové aplikace, umožňující on-line řešení vybraných úloh z oblasti Teorie systémů. Řešenými úlohami jsou jednoduché regulátory, modely hospodářské regulace, řízení s korekčními členy s stabilita systémů řešená Routh-Schurovým kritériem stability.

Včetně vytvoření skriptů v MATLABu, byly vytvořeny i související WWW stránky, poskytující uživatelům teoretický výklad k úlohám vytvořených v MWS aplikaci a je možné tyto úlohy řešit on-line. Struktura těchto WWW stránek byla záměrně rozdělena do několika samostatných bloků, aby se uživatelé mohli dobře orientovat a vše bylo přehledně uspořádáno.

Pro tvorbu WWW stránek jsem využil kaskádových stylů, pomocí nichž lze webové stránky jednoduše a efektivně upravit. Byla snaha, aby aplikace vypadala ve všech WWW prohlížečích stejně. Ne vždy se to ale podařilo. Je to způsobeno tím, že různé prohlížeče mají různě implementovány standarty W3C. Např. při nastavení šířky tabulky ve WWW stránce dochází k takové situaci, že v prohlížeči Internet Explorer 6 dochází k nesprávné interpretaci vlastnosti *width*. Do ní se započítává i vnitřní okraj tabulky (*padding*) a orámování (*border*), což se projeví na tom, že tabulka je širší jako okraje stránky prohlížeče. Tato chyba ale nebyla zjištěna v prohlížečích Mozilla Firefox. Právě v těchto dvou prohlížečích byla testována funkčnost a vzhled WWW stránek MWS aplikace. Na druhou stranu jde hlavně o designové rozdíly WWW stránek, nikterak neovlivňují funkčnost MWS aplikace.

Všechny WWW stránky naprogramovány v HTML kódu. Při vytváření webových stránek jsem využil rámců, usnadňující přehledné rozvržení webové stránky a přehlednou navigaci. Pro kontrolu zadávaných údajů ve vstupním formulář jsem využil JavaScript. K odladění funkcí JavaScriptu je možné využít nástroj JavaScript Console v Mozole, jehož

pomocí lze zjistit místo vyskytnuté chyby. V případě vypnutí JavaScriptu ve webovém prohlížeči jsou chyby ve vstupních údajích ošetřeny přímo v m-souboru.

Cíl diplomové práce byl splněn. Jednotlivé kódy v HTML i v m-souborech byly vytvořeny s důrazem na jednoduchost, aby se mohli do MWS aplikace později přidávat další úlohy.

Použitá literatura

[1] ALLEN, R.G.D. Matematická ekonomie. 1. vyd. Praha: Academica, 1971. 782 s. ISBN neuvedeno.

[2] BALÁTĚ, J. Automatické řízení. 1. vyd. Praha: BEN – technická literatura, 2003. 650 s. ISBN 80-7300-020-2.

[3] BUREŠ, V., OLEŠNICOVÁ, K. E-learning v prostředí univerzity [online]. [cit. 2008-07-13]. Dostupné na < >.

[4] KARBAN, P. Výpočty a simulace v programech Matlab a Simulink. 1. vyd. Brno: Computer Press, 2006. 224 s. ISBN 80-251-1301-9.

[5] KROPÍK, P., ŠROUBOVÁ L., VONDRÁK, M.. Užití MALTAB Database toolboxu a Web Serveru v systému ověřování znalostí studentů [online].[cit. 2008-07-15]. Dostupné na < http://dsp.vscht.cz/konference_matlab/matlab02/kropik_sroubova.pdf >.

[6] KŘUPKA, J. Teorie systémů 1. 1. vyd. Pardubice: Univerzita Pardubice, 2006. 140 s. ISBN 80-7194-923-X.

[7] NOVÁK, M. E-learning- nástroje pro tvorbu a řízení výuky [online]. [cit. 2008-07-01]. Dostupné na < http://www.volny.cz/xmichalx/bp/xnovm133_BP.pdf >.

[8] POWELL, T. A. Web design Kompletní průvodce. 1. vyd. Brno: Computer Press, 2004. 818 s. ISBN 80-722-6949-6.

[9] PUERTO, R. Remote control laboratory using Matlab and Simulink: Application to a DC motor model [online]. [cit.2008-06-01]. Dostupné na < https://isa.umh.es/arvc/documentos/articulos/IBCE04_34019594.pdf >.

[10] PUŽMANOVÁ, R. TCP/IP v kostce. 1. vyd. České Budějovice: Kopp, 2004. 607 s. ISBN 80-7232-236-2.

[11] RYBÁROVÁ, K. Tvorba e-learningovej učebnice: Návrhy regulátorov [online]. [cit. 2008-05-01]. Dostupné na <www.kirp.chtf.stuba.sk/publication_access.php?id_pub=282>.

[12] ŠVARC, I. Základy automatizace [online]. [cit. 2008-07-05]. Dostupné na < http://web.tuke.sk/sjf-kaar/stranky/Predmetove_str/TK/material/Prednasky/Zaklady_Automatizace.pdf >.

[13] THE MATHWORKS, INC. Matlab – The Langure of technical computing [online]. [cit. 2007-10-11]. Dostupné na <http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/matlab_env.pdf>

[14] THE MATHWORKS, INC. Matlab Web Server [online]. [cit. 2007-10-11]. Dostupné na <http://www.mathworks.com/products/webserver>.

[15] ZAPLATÍLEK, K.: Matlab – tvorba uživatelských aplikací. 1. vyd. Praha: Ben, 2004. 216 s. ISBN 80-7300-133-0.

Seznám obrázků

OBR. 1.1: SOUČÁSTI MWS APLIKACE. 2
OBR. 1.2: STRUKTURA SYSTÉMU MWS APLIKACE 5
OBR. 2.1: HTTP SERVER A MWS NA JEDNOM POČÍTAČI 8
OBR. 2.2: SERVER A MWS KAŽDÝ NA JINÉM POČÍTAČI 9
OBR. 2.3: MATAB WEB SERVER - PRINCIP FUNGOVÁNÍ 14
OBR. 3.1: HLÁŠENÍ O CHYBĚ PŘI NEVYPLNĚNÍ POLOŽKY VE FORMULÁŘI 17
OBR. 3.2: VÝSTUPNÍ FORMULÁŘ S CHYBOVÝM HLÁŠENÍM 20
OBR. 3.3: VLASTNOSTI BLOKU SCOPE 22
OBR. 4.1: ZPŮSOB SESTROJENÍ PŘECHODOVÉ CHARAKTERISTIKY 24
OBR. 4.2: CHARAKTERISTIKY REGULAČNÍCH ČLENŮ 24
OBR. 4.3: VSTUPNÍ FORMULÁŘ PRO MODELY JEDNODUCHÝCH REGULÁTORŮ 25
OBR. 4.4: VÝSTUPNÍ FORMULÁŘ PRO MODELY JEDNODUCHÝCH REGULÁTORŮ 26
OBR. 4.5: MATEMATICKÝ MODEL MULTIPLIKÁTORU-AKCELERÁTORU. 27
OBR. 4.6: PŘECHODOVÉ CHARAKTERISTIKY MODELŮ 28
OBR. 4.7: PŘECHODOVÉ CHARAKTERISTIKY MODELŮ 28
OBR. 4.8: VSTUPNÍ FORMULÁŘ PRO MODELY STABILIZAČNÍCH POLITIK 29
OBR. 4.9: VÝSTUPNÍ FORMULÁŘ PRO MODELY HOSPODÁŘSKÉ REGULACE 30
OBR. 4.10: STABILITA REGULAČNÍHO OBVODU 31
OBR. 4.11: JEDNODUCHÝ REGULAČNÍ OBVOD 31
OBR. 4.12: POLOHA KOŘENŮ CHARAKTERISTICKÉ ROVNICE 32
OBR. 4.13: ALGORITMUS ROUTH-SCHUROVA KRITÉRIA 33
OBR. 4.14: VSTUPNÍ FORMULÁŘ PRO VÝPOČET STABILITY SYSTÉMŮ 34
OBR. 4.15: VÝSTUPNÍ FORMULÁŘ S VÝPOČTY STABILITY SYSTÉMŮ 35
OBR. 4.16: KOREKCE ZMĚNOU ZESÍLENÍ 36
OBR. 4.17: ZAPOJENÍ SÉRIOVÉHO KOREKČNÍHO ČLENU 36
OBR. 4.18: VSTUPNÍ FORMULÁŘ PRO MODEL SÉRIOVÉHO KOREKČNÍHO ČLENU 37
OBR. 4.19: VÝSTUPNÍ FORMULÁŘ PRO MODEL SE SÉRIOVÝM KOREKČNÍM ČLENEM 38
OBR. 5.1: ROZDĚLENÍ WEBOVÉ STRÁNKY MWS APLIKACE 39
OBR. 5.2: USPOŘÁDÁNÍ HLAVNÍ ČÁSTI WEBOVÉ STRÁNKY 40
OBR. 5.3: STRUKTURA VSTUPNÍHO FORMULÁŘE 40
OBR. 5.4: MAPA STRÁNEK MWS APLIKACE 41

Seznam tabulek

TAB. 4.1: PŘENOSY REGULAČNÍCH ČLENŮ 25

Seznam příloh

Příloha 1: Obsah přiloženého CD

Příloha 2: Zdrojový kód m-souboru regulatory.m

Příloha 3: Zdrojový kód m-souboru Politiky.m

Příloha 4: Zdrojový kód m-souboru korclen.m

Příloha 5: Zdrojový kód m-souboru RSK.m

Přílohy

Příloha 1 Obsah přiloženého CD

Součástí diplomové práce je CD-ROM s následujícím obsahem:

- **HTML stránky vytvoření MWS aplikace** – zde jsou veškeré kódy a obrázky potřebné pro vytvoření stránek popisujících vytvoření MWS aplikace včetně konfiguračního souboru *matweb.conf*.
    - \index
    - \apache
    - \MWS_tvorba
- **Zdrojové kódy MWS aplikace** – zde jsou veškeré kódy HTML stránek s teoretickým výkladem úloh společně s m-soubory a simulačními modely popsaných v kap. 4 a nacházejí se v:
    - \regulatory
    - \model_MA
    - \kor_clen
    - \stab_syst

Příloha 2 – Zdrojový kód m-souboru regulatory.m

```
function HTMLout=Regulatory(in,out);
cd (in.mldir);                          %nastavení cesty pracovního adre-
sáře
mlid = in.mlid;
wscleanup('regulatorml0*.jpeg', 0.001);          %vymazání předchozího
grafu
%ověření správnosti vstupních údajů
if
(isempty(in.citatel1)|isempty(in.jmenovatel1)|isempty(in.citatel2)|isempt
y(in.jmenovatel2)|isempty(in.citatel3)|isempty(in.jmenovatel3)|isempty(in
.citatel4)|isempty(in.jmenovatel4)|isempty(in.doba))
out.hlaseni='Vyplňte špatně zadané položky ve formuláři!';
templatefile = which('error.html');
%pokud jsou vstupní údaje v pořádku, data se načtou do proměnných
else
 citatel1 = (in.citatel1);
 jmenovatel1 = (in.jmenovatel1);
 citatel2 = (in.citatel2);
 jmenovatel2 = (in.jmenovatel2);
 citatel3 = (in.citatel3);
 jmenovatel3 = (in.jmenovatel3);
 citatel4 = (in.citatel4);
 jmenovatel4 = (in.jmenovatel4);
 doba = str2num (in.doba);
 open_system ('regulator');
%nastavení parametrů jednotlivých bloků v Simulinku
 set_param('regulator/Transfer Fcn' , 'Numerator', sprintf('[%s]',
num2str(citatel1)),'Denominator',sprintf('[%s]',num2str(jmenovatel1)));
 set_param('regulator/Transfer Fcn2', 'Numerator', sprintf('[%s]',
num2str(citatel2)),'Denominator',sprintf('[%s]',num2str(jmenovatel2)));
 set_param('regulator/Transfer Fcn3', 'Numerator',
sprintf('[%s]',num2str(citatel3)),'Denominator',sprintf('[%s]',num2str(jm
enovatel3)));
 set_param('regulator/Transfer Fcn4', 'Numerator', sprintf('[%s]',
num2str(citatel4)),'Denominator',sprintf('[%s]',num2str(jmenovatel4)));
 sim('regulator',[doba]);
%vytvoření grafu ze simulovaných dat a jejich uložení do pracovního pro-
storu MATLABu
 f=figure('visible','off');
 plot(ScopeData.time, ScopeData.signals.values);
```

```
 title('Regulator');
 legend ('a)','b)','c)','d)',5)
 ylabel('Regulacní odchylka');
 xlabel('Doba simulace');
%nastaveni velikosti grafu a jeho pozice
 pos = get(gcf, 'position');
 pos(3) = 380;
 pos(4) = 310;
 set(gcf, 'Position', pos, 'PaperPosition', [1 1 13 11]);
 drawnow;
%uložení grafu v grafickém formátu *.jpeg a jeho uložení na disk
 out.GraphFileName = sprintf('regulator%s.jpeg', mlid);
 wsprintjpeg(f, out.GraphFileName);
 out.GraphFileName = sprintf('/icons/regulator%s.jpeg', mlid);
 close all;
%načtení volaného výstupního formuláře do proměnné templatefile
 templatefile = which ('regulatory2.html');
end
%zavolání výstupní šablony a odeslání výstupních dat do této šablony
HTMLout = htmlrep(out, templatefile);
end
```

Příloha 3 – Zdrojový kód m-souboru Politiky.m

```
function HTMLout=Politik(in,out);
cd (in.mldir);                          %nastavení cesty pracovního adre-
sáře
mlid = in.mlid;
wscleanup('regulatorml0*.jpeg', 0.001);          %vymazání předchozího
grafu
%ověření správnosti vstupních údajů
if(isempty(in.ZV)|isempty(in.citatel1)|isempty(in.jmenovatel1)|isempty(in
.citatel2)|isempty(in.jmenovatel2)|isempty(in.citatel)|
isempty(in.jmenovatel)|isempty(in.doba))
out.hlaseni='Vyplňte špatně zadané položky ve formuláři?!';
templatefile = which('error.html');
%pokud jsou vstupní údaje v pořádku, data se načtou do proměnných
else
 citatel = (in.citatel);
 jmenovatel = (in.jmenovatel);
 doba = str2num(in.doba);
 ZV = (in.ZV);
 citatel1 = (in.citatel1);
 jmenovatel1 = (in.jmenovatel1);
 citatel2= (in.citatel2);
 jmenovatel2= (in.jmenovatel2);
 open_system ('politiky');
%nastavení parametrů jednotlivých bloků v Simulinku
 set_param('politiky/Transfer    Fcn',    'Numerator',    sprintf('[%s]',
num2str(citatel)) ,'Denominator', sprintf('[%s]', num2str(jmenovatel)));
 set_param('politiky/Transfer    Fcn1',    'Numerator',    sprintf('[%s]',
num2str(citatel1)),'Denominator',sprintf('[%s]',num2str(jmenovatel1)));
 set_param('politiky/Transfer    Fcn2',    'Numerator',    sprintf('[%s]',
num2str(citatel2)),'Denominator',sprintf('[%s]',num2str(jmenovatel2)));
 set_param('politiky/Step','After',sprintf('[%s]',num2str(ZV)));
 sim('politiky',[doba]);
%vytvoření grafu ze simulovaných dat a jejich uložení do pracovního
%prostoru MATLABu
 f=figure('visible','off');
 plot(ScopeData.time, ScopeData.signals.values);
 grid on;
 title('Ucinky stabilizacnich politik');
 ylabel('Regulacní odchylka');
 xlabel('Doba simulace');
```

```
 legend('Bez stabilizace','Proporcionalni politika','Smisena politi-
ka',3);
%nastaveni velikosti grafu a jeho pozice
 pos = get(gcf, 'position');
 pos(3) = 380;
 pos(4) = 310;
 set(gcf, 'Position', pos, 'PaperPosition', [1 1 13 11]);
 drawnow;
%uložení grafu v grafickém formátu *.jpeg a jeho uložení na disk
 out.GraphFileName = sprintf('politiky%s.jpeg', mlid);
 wsprintjpeg(f, out.GraphFileName);
 out.GraphFileName = sprintf('/icons/politiky%s.jpeg', mlid);
 close all;
%načtení volaného výstupního formuláře do proměnné templatefile
 templatefile = which ('politiky2.html');
end
%zavolání výstupní šablony a odeslání výstupních dat do této šablony
HTMLout = htmlrep(out, templatefile);
end
```

Příloha 4 – Zdrojový kód m-souboru korclen.m

```
function HTMLout=korclen(in,out);
cd(in.mldir);                          %nastavení cesty pracovního adre-
sáře
mlid = in.mlid;
wscleanup('korclenml0*.jpeg', 0.001);          %vymazání předchozího
grafu
%ověření správnosti vstupních údajů
if(isempty(in.zesileni)|isempty(in.citatel1)|isempty(in.jmenovatel1)|isem
pty(in.citatel2)|isempty(in.jmenovatel2)|isempty(in.citatel3)|isempty(in.
jmenovatel3)|isempty(in.doba))
out.hlaseni='Vyplňte špatně zadané položky ve formuláři?!';
templatefile = which('error.html');
%pokud jsou vstupní údaje v pořádku, data se načtou do proměnných
else
 zesileni =  (in.zesileni);
 citatel1 =  (in.citatel1);
 jmenovatel1 =  (in.jmenovatel1);
 citatel2 =  (in.citatel2);
 jmenovatel2 =  (in.jmenovatel2);
 doba= str2num(in.doba);
 citatel3 = (in.citatel3);
 jmenovatel3 = (in.jmenovatel3);
 open_system ('kor_clen');
%nastavení parametrů jednotlivých bloků v Simulinku
 set_param('kor_clen/Gain', 'Gain', sprintf('[%s]',num2str(zesileni)));
 set_param('kor_clen/Transfer   Fcn',   'Numerator',   sprintf('[%s]'   ,
num2str(citatel1)),'Denominator',sprintf('[%s]',num2str(jmenovatel1)));
 set_param('kor_clen/Transfer    Fcn1','Numerator',    sprintf('[%s]'    ,
num2str(citatel2)),'Denominator',sprintf('[%s]',num2str(jmenovatel2)));
 set_param('kor_clen/Transfer    Fcn2','Numerator',    sprintf('[%s]'    ,
num2str(citatel3)),'Denominator',sprintf('[%s]',num2str(jmenovatel3)));
 set_param('kor_clen/Gain1', 'Gain', sprintf('[%s]',num2str(zesileni)));
 set_param('kor_clen/Transfer   Fcn3',   'Numerator',   sprintf('[%s]'   ,
num2str(citatel1)),'Denominator',sprintf('[%s]',num2str(jmenovatel1)));
 set_param('kor_clen/Transfer    Fcn4','Numerator',    sprintf('[%s]'    ,
num2str(citatel2)),'Denominator',sprintf('[%s]',num2str(jmenovatel2)));
 sim('kor_clen',[doba]);
%vytvoření grafu ze simulovaných dat a jejich uložení do pracovního
%prostoru MATLABu
 f=figure('visible','off');
```

```
 plot(ScopeData.time, ScopeData.signals.values);
 grid on;
 title('Korekce seriovym clenem');
 ylabel('Regulacní odchylka');
 xlabel('Doba simulace');
 legend('bez korekce','s korekci',3);
%nastaveni velikosti grafu a jeho pozice
 pos = get(gcf, 'position');
 pos(3) = 380;
 pos(4) = 310;
 set(gcf, 'Position', pos, 'PaperPosition', [1 1 13 11]);
 drawnow;
%uložení grafu v grafickém formátu *.jpeg a jeho uložení na disk
 out.GraphFileName = sprintf('korclen%s.jpeg', mlid);
 wsprintjpeg(f, out.GraphFileName);
 out.GraphFileName = sprintf('/icons/korclen%s.jpeg', mlid);
 close all;
%načtení volaného výstupního formuláře do proměnné templatefile
 templatefile = which ('korclen2.html');
end
%zavolání výstupní šablony a odeslání výstupních dat do této šablony
HTMLout = htmlrep(out, templatefile);
end
```

Příloha 5 – Zdrojový kód m-souboru RSK.m

```
function HTMLout= RSK(in, out)
%ověření správnosti vstupních údajů
if ((isempty(in.cisloA)) | (isempty(in.cisloB)))
out.hlaseni='Vyplňte špatně zadané položky ve formuláři!';
templatefile = which('error.html');
else
%pokud jsou vstupní údaje v pořádku, data se načtou do proměnných
 rovnice1 =  str2num(in.cisloA);
 rovnice2 =  str2num(in.cisloB);
 koreny1 = roots (rovnice1);
 koreny2 = roots (rovnice2);
 realne1 = real (koreny1);
 realne2 = real (koreny2);
 imag1 = imag (koreny1);
 imag2 = imag (koreny2);
%velikost vektoru
 VV1 = size (rovnice1,2);
 VV2 = size (rovnice2,2);
%počet kořenů charakteristických rovnic
 pocetKOR1 = VV1 -1;
 pocetKOR2 = VV2 -1;
 n1 = 1;
 n2 = 1;
 x1 = realne1(n1);
 x2 = realne2(n2);
%výpočet stability pro první rovnici
  while and((n1<pocetKOR1),(x1<0)),
    n1=n1+1;
    x1 = realne1(n1);
  end
  if realne1(n1) < 0
    vystup1='Systém je stabilni';
  else
    vystup1='Systém je nestablni';
  end
%výpočet stability pro druhou rovnici
  while and((n2<pocetKOR2),(x2<0)),
    n2=n2+1;
    x2 = realne2(n2);
  end
```

```
  if realne2(n2) < 0
    vystup2='Systém je stabilni';
  else
    vystup2='Systém je nestabilni';
  end
%uložení vypočítaných dat do výstupních údajů
 out.realne1 =(realne1);
 out.realne2= (realne2);
 out.imag1 = (imag1);
 out.imag2 = (imag2);
 out.system1 =(vystup1);
 out.system2 = (vystup2);
%načtení volaného výstupního formuláře do proměnné templatefile
 templatefile = which('RSK2.html');
end
%zavolání výstupní šablony a odeslání výstupních dat do této šablony
HTMLout = htmlrep(out, templatefile);
end
```