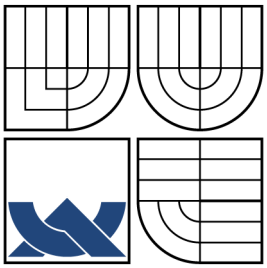
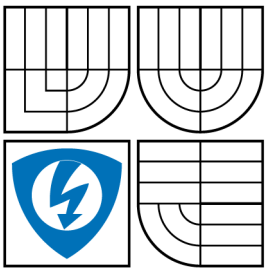


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

MOŽNOSTI SPOLUPRÁCE PROSTŘEDÍ MATLAB S DALŠÍMI PROGRAMY A PROGRAMOVACÍMI JAZYKY

MATLAB ENVIRONMENT POTENTIAL FOR COOPERATION WITH OTHER APPLICATIONS AND PROGRAMMING LANGUAGES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JIŘÍ ŠTEFEK
AUTHOR

VEDOUCÍ PRÁCE — ING. HICHAM ATASSI
SUPERVISOR

BRNO 2009

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Jiří Štefek	***ID:***	78724
Ročník:	3	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Možnosti spolupráce prostředí Matlab s dalšími programy a programovacími jazyky

POKYNY PRO VYPRACOVÁNÍ:

Vytvořte v prostředí Matlab serverovou aplikaci, která by umožňovala hromadné zpracování multimediálních souborů. Aplikace by měla poskytovat uživateli možnost dálkově spravovat server a využívat poskytované funkce pro zpracování souborů. Aplikace by měla být schopna exportovat výsledky zpracování do souborů různého formátů.

DOPORUČENÁ LITERATURA:

[1] Karban P., Výpočty a simulace v programech Matlab a Simulink. Computer press 2006.
[2] Zaplatílek K.Doňar B., Matlab-tvorba uživatelských aplikací, nakladatelství BEN 2005.
[3] Zaplatílek K.Doňar B., Matlab-začínáme se signály, nakladatelství BEN 2007.

Termín zadání:	9.2.2009	***Termín odevzdání:***	2.6.2009

Vedoucí práce: Ing. Hicham Atassi

prof. Ing. Kamil Vrba, CSc.
předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato práce je zaměřena na možnosti prostředí Matlab. Možnosti ve smyslu sdílení nebo zpracování dat mezi Matlabem a jinými programovacími jazyky a programy. Práce si klade za cíl vytvořit návod a přehled v těchto možnostech a dále zviditelňuje možnosti prezentace dat přes webové stránky.

KLÍČOVÁ SLOVA

Matlab, programovací jazyk, program, Excel, DDE, FTP, C, Java, MSP, JSP.

ABSTRACT

This thesis is focused on possibillities of Matlab enviroment. Mainly shows the possibillities of sharing and processing data between Matlab and other programs and programming languages. The main goal is to create an introducement and overview in these possibillities and shows potentialities to present data through web sites.

KEYWORDS

Matlab, programming language, program, Excel, DDE, FTP, C, Java, MSP, JSP.

ŠTEFEK J. *Možnosti spolupráce prostředí Matlab s dalšími programy a programovacími jazyky*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 66 s. Vedoucí bakalářské práce Ing. Hicham Atassi.

Poděkování

Hluboce děkuji vedoucímu bakalářské práce Ing. Hichamu Atassimu, za důkladné a věcné konzultace, rady při zpracování bakalářské práce a vždy trpělivý přístup. Dále děkuji všem lidem za rady a opravy při zpracování této práce.

V Brně dne: . . . . . .

. . . . . .

(podpis autora)

OBSAH

Úvod **10**

1 Spolupráce s jinými programovacími jazyky **11**
1.1 Úvod . . . 11
1.2 Java a Matlab . . . 11
1.2.1 Úvod . . . 11
1.2.2 Ukázka . . . 11
1.3 C, C++, Fortran a Matlab . . . 12
1.3.1 Úvod . . . 12
1.3.2 Využití funkce napsané v jazyku C v Matlabu. . . . 13
1.3.3 Využití Matlabu v programu napsaném v jazyku C . . . 14

2 Získávání dat z externích zdrojů **17**
2.1 Úvod . . . 17
2.2 Komunikace přes RS-232 . . . 17
2.2.1 Objekt sériového portu . . . 17
2.2.2 Připojení objektu . . . 18
2.2.3 Zápis řídícího znaku . . . 18
2.2.4 Příjem dat . . . 18
2.2.5 Zrušení spojení . . . 19
2.3 Komunikace pomocí FTP . . . 20
2.3.1 Vytvoření FTP objektu . . . 20
2.3.2 Přenos a manipulace s daty . . . 20
2.3.3 Konec spojení . . . 20
2.3.4 Jednoduchý příklad . . . 21
2.4 Dynamická výměna dat v OS Windows . . . 21
2.4.1 Vytvoření kanálu pro komunikaci s aplikací . . . 21
2.4.2 Přenos dat . . . 22
2.4.3 Ukončení spojení . . . 22
2.4.4 Jednoduchý příklad . . . 23

3 Matlab Server Pages **24**
3.1 Úvod . . . 24
3.2 Konfigurace MSP . . . 24
3.2.1 Úvod . . . 24
3.2.2 Instalace . . . 25
3.2.3 Nastavení . . . 25

3.3 Jednoduchá aplikace . . . 27
3.3.1 Zvolený program a základní princip . . . 27
3.3.2 Co se skrývá ve zdrojovém kódu . . . 28
3.4 Aplikace č.1 . . . 29
3.4.1 Úvod . . . 29
3.4.2 Program . . . 29

4 Aplikace č.2 **32**
4.1 Úvod . . . 32
4.2 Spuštění . . . 32
4.3 Systematizace souborů v projektu . . . 33
4.4 Uživatelská práva . . . 36
4.5 Možnosti . . . 36
4.6 Ukázka aplikace . . . 37
4.7 Testování . . . 44

5 Závěr **46**

Literatura **47**

Seznam symbolů, veličin a zkratek **49**

Seznam příloh **50**

A První příloha **51**
A.1 Funkce v jazyku C a jeho implementace pro Matlab . . . 51
A.2 Volání Matlabu z jazyka C . . . 53

B Druhá příloha **55**
B.1 MSP příkazy . . . 55
B.2 Program - Přesměrování . . . 56
B.3 Program - Upload Souboru . . . 56
B.4 Program - UploadBean . . . 57
B.5 Program - Přesměrování pomocí dokumentu XML . . . 58
B.6 Program - Výběr analýzy . . . 59
B.7 Program - Výpis výsledků . . . 61
B.8 Program - MFile s analýzou . . . 63

C Třetí příloha **66**
C.1 Obsah CD . . . 66

SEZNAM OBRÁZKŮ

1.1 Ukázka programu *vyresip*. 12
1.2 Volání funkce napsané v jazyku C z Matlabu. 14
1.3 Volání Matlabu z jazyka C – graf. 16
1.4 Volání Matlabu z jazyka C – okénko s proměnnými. 16
2.1 FTP v Matlabu s podrobnostmi. 22
2.2 DDE a Matlab – příkazová řádka s podrobnostmi a výsledek uložený v Excelu. 23
3.1 MSP – Nově vytvořené ikony na ploše. 25
3.2 MSP – Cesta k obrázkům a přijatým souborům. 26
3.3 MSP – Nastavení cesty k Matlabu. 26
3.4 MSP – Nastavení cesty k JSDK. 26
3.5 MSP – Ukázka jednoduchého požadavku. 28
3.6 MSP – Ukázka jednoduché odezvy. 28
3.7 MSP – Program č.1 – Formulář na upload souboru a Formulář s volbou analýz. 30
3.8 MSP – Program č.1 – Výsledek. 31
4.1 Aplikace č.2 – Systém souborů. 34
4.2 Aplikace č.2 – Info stránka. 37
4.3 Aplikace č.2 – Registrace. 38
4.4 Aplikace č.2 – Registrace 3. 38
4.5 Aplikace č.2 – Přihlášený. 39
4.6 Aplikace č.2 – Nahrávání souborů. 40
4.7 Aplikace č.2 – Výběr analýzy. 40
4.8 Aplikace č.2 – Výsledky analýzy. 41
4.9 Aplikace č.2 – Výsledek analýzy – obrázek. 41
4.10 Aplikace č.2 – Výsledek analýzy – text. 42
4.11 Aplikace č.2 – Úprava analýzy. 43
4.12 Aplikace č.2 – Zátěžový test CPU. 44
4.13 Aplikace č.2 – Zátěžový test paměti. 45
4.14 Aplikace č.2 – Zátěžový test paměti 2. 45

SEZNAM TABULEK

1.1 Tabulka s možnými příkazy pro Matlab. 15
2.1 Tabulka s parametry spojení přes sériový port. 18
2.2 Tabulka příkazů pro manimpulaci se soubory a adresáři přes FTP. . . 21
4.1 Tabulka s významy složek aplikace č.2. 35
4.2 Tabulka s významy uživatelských práv aplikace č.2. 36

ÚVOD

Tato práce je zaměřena na prostředí Matlab a na jeho možnosti v oblasti spolupráce s ostatnímy programy, programovací jazyky a periferiemy.

Matlab je výkonné programové prostředí a taktéž skriptovací jazyk. Stále více se využívá ve vědocko-technické oblasti. Spektrum jeho schopností zahrnuje např. numerické výpočty, analýzu dat, zpracování signálů, modelování, řešení diferenciálních rovnic atd. Více o možnostech tohoto programu viz [7], [8]. Název MATLAB vznikl zkrácením slov MATrix LABoratory, nebo-li maticová laboratoř, což odpovídá skutečnosti, že klíčovou datovou strukturou při výpočtech v MATLABu jsou matice.

První část je zaměřená na spolupráci tohoto prostředí s programovacími jazyky. Důkladněji jsou zde probrány možnosti spolupráce s jazyky Java a C, konkrétně volání programu napsaném v těchto jazycích z prostředí Matlab a opačná komunikace, tedy využití schopností tohoto programu v programech napsaných těmito programovacími jazyky.

Druhá část práce je zaměřena na spolupráci s periferiemy, ostatnímy programy a komunikaci pomocí protokolů pro přenos dat. K naleznutí je zde postup pro navázání komunikace přes seriovou linku, přenos dat přes FTP a dynamická výměna dat mezi prostředím Matlab a programem Microsoft EXCEL.

Třetí část je věnována opensource projektu MSP (Stránky Matlab-Serveru – Matlab Server Pages) a jeho možnostem. MSP umožňuje i mírně pokročilým začátečníkům v oblasti vývoje programů ovládnout nástroj pro prezentaci dat získaných z prostředí Matlab na webových stránkách.

Čtvrtá část je zaměřena na výsledný program spojující nabyté zkušenosti. Byla vytvořena webová aplikace na bázi JSP stránek. Aplikace umí předávat a prezentovat výsledky zaslané a získané z Matlabu. Je zde možnost manipulace souborů na vzdáleném serveru a funkce pro spravování uživatelských účtů. Tento oddíl zahrnuje postup instalace, vysvětlení základních algoritmů a fází programu a vysvětluje jeho možnosti a využití.

1 SPOLUPRÁCE S JINÝMI PROGRAMOVACÍMI JAZYKY

1.1 Úvod

Vývojové prostředí Matlab umí používat, komunikovat a spolupracovat s programy vytvořenými v programovacích jazycích jako je C, C++, Fortran nebo Java. V následujících kapitolách jsou uvedeny některé z vypsaných možností.

1.2 Java a Matlab

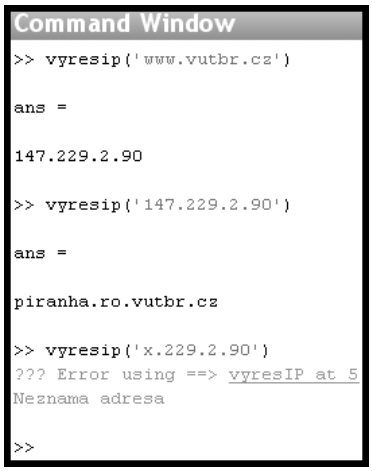
1.2.1 Úvod

Ke spuštění programů napsaných v Javě je nutností mít nainstalovaný některý z JVM(Java Virtuální stroj – Java Virtual Machine), který zajišťuje vazbu na hardware konkrétního PC a interpretuje Javovský bajtkód. JVM je obsažen v každém JRE(Prostředí pro spouštění Java programů – Java Runtime Enviroment), nebo v komplexnějším JSDK(Prostředí pro vývoj Java programů – Java Software Development Kit). Tato prostředí jsou volně k stažení na adrese viz [14]. K opačnému přístupu, k Matlabu z programů v Javě je nutnost mít požadované knihovny pro takovou práci, např. [5] nebo [15] . Na tuto druhou variantu jsou aktivně zaměřeny kapitoly 3 a 4.

1.2.2 Ukázka

V tomto oddíle je popsán příklad, kde Matlab využívá funkcí z knihoven jazyka Java. Přístup k programům v Javě je jednoduchý. Následující příklad znázorňuje vytvoření Java objektu, která má přístup k metodě, která umí zpracovat IP adresu a vrátí jeho DNS jméno, nebo naopak. Jde o metody třídy *java.net.InetAddress*.

Následující funkce **vyresip** očekává na vstupu řetězec znaků, který představuje buď IP adresu, nebo jméno DNS. Z proměnné **adresa** je vytvořen objekt a pomocí jeho metod je do proměnných hostname a ipadress uložena IP adresa a název DNS. Následují rozhodovací podmínky, díky kterým je vrácena požadovaná hodnota – DNS jméno, v případě zadané IP a naopak. Ošetření neexistující adresy je provedeno pomocí **try** a **catch**. V případě neexistující adresy se vytiskne chyba a text **Neznama adresa**. Ověření funkčnosti programu viz obr. 1.1.



Obr. 1.1: Ukázka programu *vyresip*.

Zdrojový kód programu vyresip:

```
%program vyresip
function x=vyresip(input)
try
    address = java.net.InetAddress.getByName(input);
catch
    error('Neznama adresa');
end
hostname = address.getHostName;
ipaddress = address.getHostAddress;
if strcmp(input,ipaddress)
    x=hostname;
else
    x=ipaddress;
end;
```

1.3 C, C++, Fortran a Matlab

1.3.1 Úvod

Z příkazového řádku prostředí Matlab lze volat vlastní funkce napsané v programovacích jazycích C,C++ nebo Fortran, přitom se k nim Matlab bude chovat jako k vlastním funkcím. K docílení takového stavu je nutno převést tyto soubory/funkce do takzvaných MEX-souborů. Zkratka MEX vyjadřuje Matlab EXecutable, tedy jde o soubory spustitelné Matlabem. Výrazně se však doporučuje omezení volání takto

vytvořených funkcí kvůli časové náročnosti. V tomto oddíle je zveřejněn jen postup při volání funkce napsané v jazyce C. Procedura u ostatních programovacích jazyků je velice podobná.Více o spoluprácí s ostatnímy programovacími jazyky i o informacích viz [8].

Opačná komunikace, tedy využití Matlabu a jeho funkcí v programech napsaných výše zmíněnými programovacímy jazyky samozřejmě také lze docílit. Pro představu: zavolání Matlabu pro výpočet transpozice matice nebo komplexnějších funkcí jako je například FFT(Rychlá Fourierova Transformace – Fast Fourier Transformation).

1.3.2 Využití funkce napsané v jazyku C v Matlabu.

Uložení funkce do pracovního adresáře

První možností je nahrát **.c** soubor s funkcí do aktuálního pracovního adresáře Matlabu. Druhou možností je změnit aktuální adresář Matlabu k požadované funkci.

Tento krok je nutný pro volání kompilátoru, je samozřejmě možné ho provést i později, avšak pro pořádek je doporučeno ho provést už teď.

Vytvoření brány mezi Matlabem a funkcí

Do zdrojového kódu s funkcí se přidá hlavičkový soubor pro komunikaci s Matlabem **include "mex.h"**. A vytvoří se tzv. bránová funkce s názvem `mexFunction`. Jde o funkci, která se zavolá po spuštění MEX souboru a zajišťuje předávání parametrů z Matlabu k funkci, ošetření chybných vstupů a formát výstupu. Dalo by se říct, že se jedná o zastoupení funkce **Main** známé z jazyka C.

Kompilace

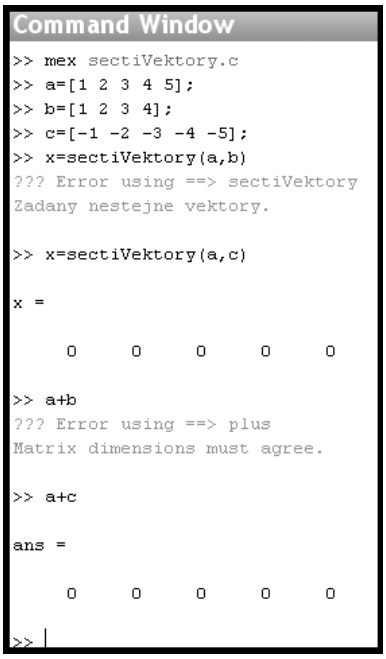
Funkci obohacenou o bránovou funkci je nutné zkompilovat pro další využití. Kompilace se provede z příkazové řádky Matlabu, příkazem:

```
mex nazevSouboruSFunkci.c
```

Při první kompilaci Matlab na tuto událost zareaguje vyvoláním nabídky s možností výběru kompilátoru. Dále se na tuto volbu neodkazuje. Funkci ze zkompilovaného souboru lze nyní běžně používat z příkazové řádky Matlabu, ovšem musí se nalézat v aktuálním pracovním adresáři.

Ukázka

Vytvořený program umí sečíst dva vektory. Jsou v něm ošetřeny chybné události, jako např. zadání dvou rozdílně dimenzovaných vektorů. Okomentovaný zdrojový



Obr. 1.2: Volání funkce napsané v jazyku C z Matlabu.

kód je umístěn v příloze A.1. Ukázka kompilace a funkčnosti programu je uvedena na obr. 1.2.

1.3.3 Využití Matlabu v programu napsaném v jazyku C

Princip

Při psaní programu můžeme volat prostředí Matlab pro výpočty přes soubor funkcí, které jsou uvedeny v tab. 1.1. Přes tyto příkazy přistupujeme k příkazové řádce Matlabu a k jeho proměnným. Díky těmto jednoduchým funkcím lze implementovat jakýkoliv program nebo funkci.

Uložení programu do pracovního adresáře

Postup je stejný jako v předchozí kapitole 1.3.2. Tento krok je nutný udělat opět kvůli bezproblémové kompilaci.

Kompilace

Vytvořený soubor umístěný v pracovním adresáři se zkompilujeme pomocí MEX kompilátoru s přepínačem **"-f"**, který slouží pro využití kompilace externím souborem. Soubor se zvolí **lccengmatopts.bat** [1] , který zahrne do procesu potřebné knihovny pro práci s Matlabem a výsledek se zkompiluje pomocí kompilátoru **lcc**.Více

[1] lcc – kompilátor, eng – engine, matopts – mathematic operations – matematické operace

Tab. 1.1: Tabulka s možnými příkazy pro Matlab.

funkce	význam
engOpen	Otevře příkazovou řádku MATLABu
engClose	Ukončí příkazovou řádku MATLABu
engGetVariable	Získá proměnnou z MATLABu
engPutVariable	Vloží proměnnou ke zpracování do MATLABu
engEvalString	Spustí příkaz v MATLABu
engOutputBuffer	Vytvoří buffer k uložení výstupního textu z Matlabu
engOpenSingleUse	Otevře příkazovou řádku MATLABu pro jedno, nesdílené použití
engGetVisible	Zjistí viditelnost příkazové řádky
engSetVisible	Nastavení viditelnosti příkazové řádku MATLABu

o kompilátoru a jeho licenci viz [6]. Po kompilaci vznikne v aktuálním pracovním adresáři stejnojmenný soubor, tentokrát se spustitelnou příponou **.exe**.
Celý příkaz:

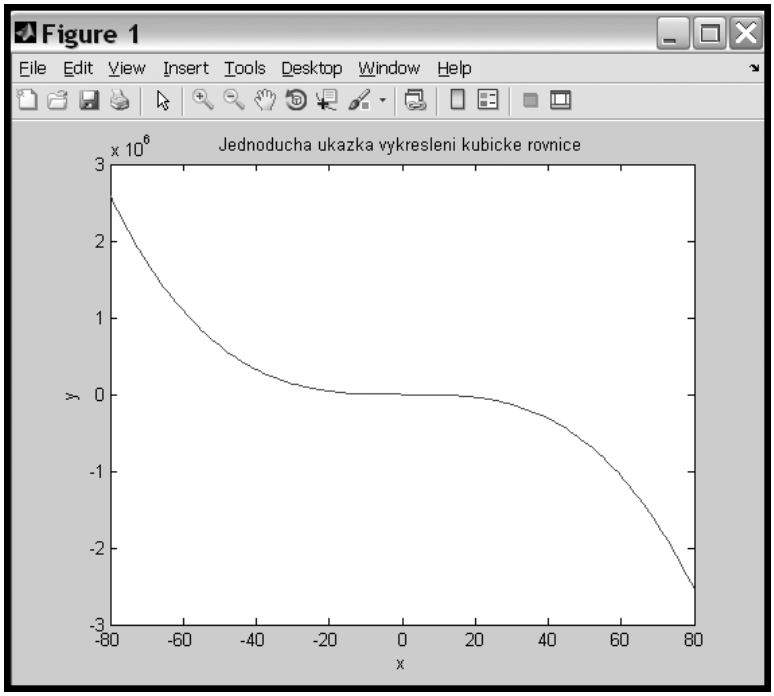
```
mex -f lccengmatopts.bat soubor.c
```

Ukázka

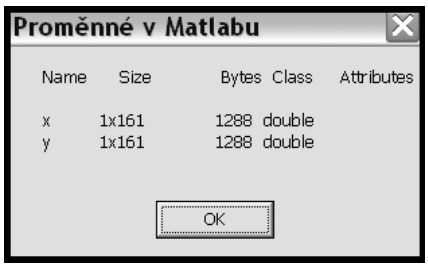
Vytvořená ukázka vypočte a vykreslí kubickou funkci, z rovnice (1.1) ve zvoleném rozsahu hodnot od -80 do 80.

$$y = -5 \cdot x^3 + 3 \cdot x^2 + x. \tag{1.1}$$

K výpočtu i vykreslení výsledku se využije výkonné prostředí Matlab. Výstupem bude graf a okénko se seznamem proměnných uložených v Matlabu. Po odkliknutí tlačítka **OK** se uzavře příkazová řádka Matlabu i vykreslený graf. Zdrojový kód je umístěn v příloze A.2. Vytvořený graf a okno s proměnnými na obrázcích 1.3 a 1.4.



Obr. 1.3: Volání Matlabu z jazyka C – graf.



Obr. 1.4: Volání Matlabu z jazyka C – okénko s proměnnými.

2 ZÍSKÁVÁNÍ DAT Z EXTERNÍCH ZDROJŮ

2.1 Úvod

Tato část se zabývá získávání dat z externích zařízení a zdrojů. Pro tuto kapitolu budou za externí zdroje dat považovány data jiných programů než prostředí Matlab. Jako zařízení lze uvažovat např. multimetr, který měří elektrické napětí, jiným zdrojem může být výsledek výpočtu zpracovaný na vzdáleném počítači, atd. .

Samozřejmě zde není prostor k výčtu všech možností prostředí Matlab, ale jsou zde uvedeny jen některé okruhy, ale zato vystiženy detailně.

2.2 Komunikace přes RS-232

Přenos mezi dvěma entitami může být buď synchronní, který je vhodnější pro větší objemy přenášených dat, nebo asynchronní, který je vhodnější při delším vedení. Komunikace pomocí RS-232, znamená komunikaci přes sériový port, tedy sériovým způsobem přenosu dat. Přes toto rozhraní lze přenášet data např. mezi PC a měřícím přístrojem. Komunikace s prostředím Matlab se dá popsat v těchto fázích:

1. vytvoření objektu sériového portu
2. připojení objektu sériového portu k přístroji
3. zápis řídících znaků do paměti přístroje
4. přijmutí dat
5. zrušení spojení.

2.2.1 Objekt sériového portu

Prvním krokem je vytvoření objektu sériového portu. K vytvoření takového objektu je použit příkaz:

```
serial(parametry)
```

Výčet všech jeho možných parametrů, je k nalezení v tab. 2.1 . Kompletní příkaz pro vytvoření objektu sériového portu bude vypadat například takto:

```
spojeni=serial('COM1','baudrate',9600,'databits',8,
               'parity','none','stopbits',1,'CR')
```

V tomto případě půjde o komunikaci na portu COM1 s přenosovou rychlostí 9600 (bit/s) , bitů s daty bude 8, bez parity, 1 stopbit a data se budou přijímat. Navíc se spojení, jakožto objekt, ještě musí uložit do proměnné, v tomto případě **spojeni**, pro budoucí přístup k metodám tohoto objektu.

Tab. 2.1: Tabulka s parametry spojení přes sériový port.

parametr	možné nastavení
port	*COM1*, *COM2...*
baudrate (přenosová rychlost)	*115200*, *57600*, *38400*, *19200*, *9600*, *4800*,*2400* ,*1200* , *300*, nebo *110*
databits (počet data bitů)	*7* nebo *8*
stopbits (počet stop bitů)	*1* nebo *2*
parity (parita)	*none*, *odd* nebo *even* (žádná, sudá nebo lichá)
terminator	*CR nebo LF* (CR pro příjem dat, LF odesílání)

2.2.2 Připojení objektu

S vytvořeným objektem sériového portu nyní lze manipulovat. K otevření komunikace slouží příkaz:

```
fopen(objekt)
```

Parametrem bude objekt vytvořený v předchozí fázi uložený do proměnné **spojeni**. Tímto zápisem je objekt připojen:

```
fopen(spojeni)
```

2.2.3 Zápis řídícího znaku

Každý přístroj, který je vybavený RS232, by měl mít ve svém manuálu podrobně popsáno, jaký řídící znak vyvolá odezvu přístroje např. pro odeslání dat. Pro názornost je zde použit řídící znak **'X'**, který v tomto ukázkovém případě vyvolá odeslání obsahu displeje. Zápis řídícího znaku do přístroje provedeme pomocí příkazu:

```
fprintf(objekt, ridiciZnak)
```

Z minulého a předminulého bodu je již otevřeno spojeni na portu COM1 a informace uložené v proměnné **spojeni**. Tedy parametry jsou:

```
fprintf(spojeni, 'X')
```

2.2.4 Příjem dat

Při správném postupu lze nyní přijmout data pomocí příkazu:

```
fscanf(objekt)
```

Vstupnímy parametry této funkce bude opět objekt uložený v proměnné **spojeni**, navíc je vhodné si data uložit do nové proměnné, která zde nese název **prijataData**:

```
prijataData=fscanf(spojeni)
```

Získané data se ukládají do pole znaků . Přístup k nim je podobný jako u jiných polí. V tomto případě je přijato pole o rozměru 9 znaků.

```
prijataData=0.12345A~
```

K převedení přijatého čísla do zpracovatelnější podoby použijeme příkaz **str2num**, který převede řetězec na číslo. Pro přístup k užitné hodnotě, v tomto případě zanedbáním hodnoty veličiny – **0.12345**, provedeme:

```
pouzitelnaData=str2num(prijataData(1:7))
```

Proces příjem dat lze samozřejmě opakovat vícekrát. Tuto proceduru je proto vhodné uložit do M-File a podle potřeby ji volat, například s opakováním po určité časové prodlevě a zautomatizovat si tímto měření.

2.2.5 Zrušení spojení

Tento krok by se dal přirovnat k situaci jako je tomu například s práci se soubory v jazyku Java nebo i C, kde musí se otevřený soubor, v našem případě spojení, uzavřít, aby k němu mohli přistupovat i jiné programy. Uzavření objektu pro sériovou komunikaci lze dosáhnout:

```
fclose(parametr)
```

V uvedeném názorném případě tomu je:

```
fclose(spojeni) .
```

Nyní je spojení nadobro uzavřeno a nelze s ním už manipulovat. Pro novou komunikaci je nutno znovu port otevřít, viz předešlý postup od 2.2.1. Více informací a příkladů k tomuto tématu viz [11]. Pro více informací o adresování a práci s maticemi v Matlabu a rovněž i funkci **str2num** viz [17].

2.3 Komunikace pomocí FTP

FTP je komunikační protokol určený pro přenos a manipulaci s daty z a na vzdáleném počítači. Na vzdáleném počítači však musí běžet FTP server, na kterém jsou spravovány uživatelské účty, hesla k jejich přístupu a práva jejich možností. Práva mohou být k čtění, zápisu, tvoření a přidávání složek nebo souborů a k jejich mazání. Tento protokol je výhodný pro přenos většího objemu dat. Komunikace s prostředím Matlab se dá popsat v těchto fázích:

1. vytvoření FTP objektu
2. přenos a manipulace s daty
3. konec spojení.

2.3.1 Vytvoření FTP objektu

Pro zahájení komunikace potřebujeme příkaz **ftp(adresa)**. Pro uložení komunikačního kanálu nám bude sloužit proměnná **spojeni** a jako parametry použijeme textový řetězec s adresou FTP serveru a textové řetězce s jménem a heslem pro přístup. V případě anonymního připojení použijeme

```
spojeni=ftp('testovaciFTPserver.cz')
```

V opačném případě:

```
spojeni=ftp('testovaciFTPserver.cz','uzivatelskeJmeno','heslo')
```

V kladném případě Matlab vypíše hlášení o vytvoření FTP objektu, v opačném případě chybové hlášení.

2.3.2 Přenos a manipulace s daty

S vytvořeným objektem lze nyní manipulovat pomocí příkazů viz tab. 2.2.

2.3.3 Konec spojení

Jak již bylo zmíněno v předchozí kapitole, je nutnost za sebou zavřít spojení.

```
close(FTPobjekt)
```

V uvedeném případě tedy:

```
close(spojeni)
```

Tab. 2.2: Tabulka příkazů pro manimpulaci se soubory a adresáři přes FTP.

funkce	význam
ascii	nastavení FTP pro přenos textových souborů
binary	nastavení FTP pro binární přenos (jiné než textové soubory)
cd	změna současného adresáře
delete	vymazání souboru na FTP serveru
dir	výpis obsahu aktuálního adresáře
mget	stahování souboru ze serveru
mkdir	vytvoření adresáře na serveru
mput	nahrání souboru na server
rename	přejmenování souboru na serveru
rmdir	vymazání adresáře na serveru

2.3.4 Jednoduchý příklad

V tomto oddíle je ukázka jednoduchého sledu příkazů pro navázání FTP spojení a získání souboru **dulezitaInformace.x** ze serveru **192.168.14.66** na portu **6996**. Zdrojový kód příkladu:

```
spojeni=ftp('192.168.14.66:6996')
mget(spojeni,'dulezitaInformace.x')
close(spojeni)
```

Samozřejmě na každý příkaz Matlab zareaguje, podrobnosti na obr. 2.1. Další informace a příklady k nalezení v [11].

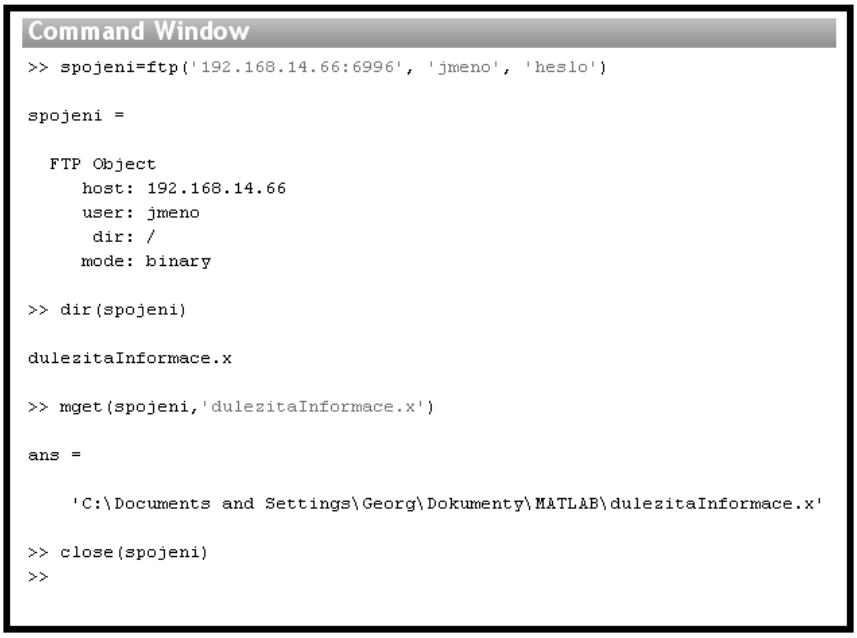
2.4 Dynamická výměna dat v OS Windows

DDE (Dynamická výměna dat – Dynamic data exchange) znamená, že lze pracovat s daty otevřených programů a to obousměrně, tedy Matlab $\Leftrightarrow$ Program. Komunikace s prostředím Matlab se dá popsat v těchto fázích:

1. vytvoření kanálu pro komunikaci s aplikací
2. přenos dat
3. ukončení spojení.

2.4.1 Vytvoření kanálu pro komunikaci s aplikací

Pro vytvoření komunikace potřebujeme příkaz **ddeinit(parametry)**. Pro uložení kanálu pro komunikaci bude sloužit proměnná **spojeni** a jako parametry budou



Obr. 2.1: FTP v Matlabu s podrobnostmi.

použity textový řetězec s názvem spuštěné aplikace a textový řetězec určující soubor otevřený aplikací. Jediná zdokumentovaná komunikace je Matlab $\Leftrightarrow$ Excel, proto bude středem tohoto oddílu. Příkaz pro navázání spojení s programem Excel, který pracuje s otevřeným souborem **soubor.xls**:

```
spojeni=ddeinit('excel','soubor.xls')
```

2.4.2 Přenos dat

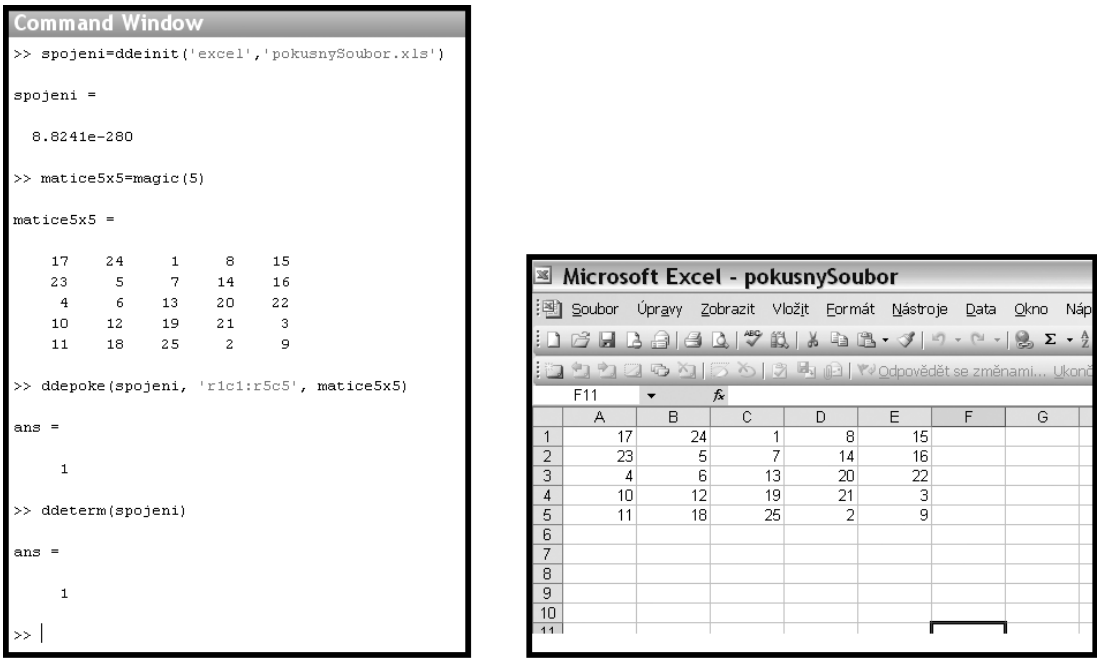
Pro přijímání dat slouží příkaz **ddereq(parametry)**, naopak pro odesílání **ddepoke(parametry)**. Následně uvedený příklad přijme data z předešle vytvořeného spojení. Konkrétně přijme údaj uvedený v 1. řádku 1. sloupce a uloží ho do proměnné **prijataData**.

```
prijataData=ddereq('spojeni','r1c1')
```

2.4.3 Ukončení spojení

Spojení ukončíme pomocí funkce **ddeterm(parametr)**. V souvislosti z předešle vytvořeném spojení:

```
ddeterm('spojeni')
```



Obr. 2.2: DDE a Matlab – příkazová řádka s podrobnostmi a výsledek uložený v Excelu.

2.4.4 Jednoduchý příklad

Uvedený příklad provede uložení matice rozměrů **5x5** vytvořené v Matlabu pomocí funkce **magic(5)** do otevřeného Excelovského souboru nesoucí příznačný název **pokusnySoubor.xls**. Zdrojový kód příkladu:

```
spojeni=ddeinit('excel','pokusnySoubor.xls')
ddepoke(spojeni, 'r1c1:r5c5')
ddeterm('spojeni')
```

Podrobnosti a výsledky lze vidět na obr. 2.2. Více informací a příkladů k nalezení např. v [11].

3 MATLAB SERVER PAGES

3.1 Úvod

K tvorbě aplikace komunikující na bázi klient-server, kde si klient vyžádá přes webové rozhraní zpracování zadaných dat a server vybaven [1] MATLABem jej provede a vrátí výsledek, je balíček MSP velice výhodný. Umožňuje skládat dohromady předem vytvořené moduly, nebo přesněji řečeno značky(tagy), a vytvořit tak téměř všehoschopnou aplikaci. Samozřejmě konstrukce vlastních tagů nebo i jiných pomocných podprogramů je rovněž podporována, ovšem vyžaduje jisté znalosti programování JSP(Webové stránky na bázi Javy – Java Server Pages). Více o programování JSP stránek viz [4] nebo [3]. Protože toto programování vychází více méně z programování v jazyce Java, potřebné informace o tomto programování viz [16] a [2].

MSP(Stránky Matlab-Serveru – Matlab Server Pages) je opensource projekt, nebo-li volně šířitelný projekt. Umožňuje vývojářům jakékoliv využití a vylepšení. Narozdíl od projektů *MATLAB Web Server*, který od verze Matlab 2006b, již není dostupný, nebo *The Ohio State University – "A Java Based Web Interface to MATLAB"* jde o nekomerční a veřejně použitelný produkt.

Navíc umožňuje rozložené zpracování výpočtu nebo souběžného zavolání aplikace na více PC díky vzdálenému volání procedur a jejich přenášení pomocí http protokolu. Je zde implementováno i vzdálené spouštění Matlabu, tedy na serveru s MSP nemusí být Matlab nainstalován a díky webovým službám s ním lze vzdáleně komunikovat. Navíc je zde vytvořeno i prostředí pro komunikaci s databázemi. Veškerá vzájemná komunikace a výměna dat mezi MSP a Matlab je zprovozněna pomocí knihovny JMatlink, viz [15].

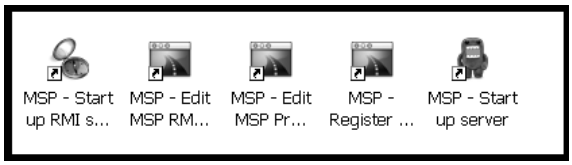
3.2 Konfigurace MSP

3.2.1 Úvod

Balíček MSP obsahuje opensource programy Apache Tomcat Web Server, který slouží jako kontejner pro veškeré JSP stránky a jejich pomocné objekty [2] a samozřejmě jako webový server. Více o Apache Tomcat Web Serveru na [1].

[1] MSP dokonce obsahuje i metody pro volání Matlabu ze vzdáleného PC, takže webový server nemusí ani Matlab obsahovat

[2] například objekty typu JavaBeans



Obr. 3.1: MSP – Nově vytvořené ikony na ploše.

3.2.2 Instalace

Na stránkách [9] je volně dostupný instalační balíček pro OS Windows. Je doporučeno dbát na umístění nainstalované aplikace, protože jejich pozice bude potřeba pro další nastavení. Navíc se v cestě **nesmí objevit mezera**.

Další náležitostí je mít nainstalované minimálně Java 2 SDK(Prostředí pro vývoj Java programů – Java Software Development Kit) verze 1.4.2.08 . Volně dostupná na [14]. V tomto návodu je pro jednoduchost vše nainstalováno na: *disk D:/.*

3.2.3 Nastavení

Po nainstalování potřebných komponent [3] je čas na nastavení. Na obr. 3.1 jsou znázorněny ikony vytvořené při instalací MSP. V této sekci bude probírán pouze cyklus nastavení, kdy server s MSP bude zároveň obsahovat i Matlab. Pro nastavení vzdáleného přístupu k Matlabu viz tutoriály na [9]

Konfigurace MSP.properties

Prvním úkolem je nastavit cestu k nainstalovanému balíčku MSP. Označením zástupce **MSP - Edit MSP Properties** a **upravením** se zobrazí otevřený soubor v **Poznámkovém Bloku**. Střed zájmu bude tvořit první řádek kde je uvedeno „tovární“ nastavení cesty k nainstalovaným MSP. V uvedeném případě, kdy je vše nainstalované v **root adresáři disku D**, se musí nastavit na :

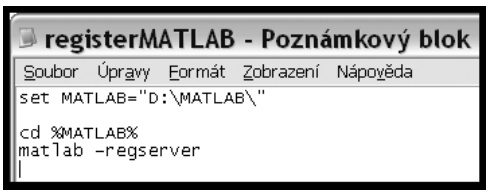
```
set INSTALLATION_DIR=d:\MATLABServerPages
```

Druhým úkolem je úprava **MSP.properties**, kde se nastaví cesta k adresáři, kde se budou uchovávat přijaté soubory a obrázky. Pro úpravu těchto vlastností stačí spustit zástupce **MSP - Edit MSP Properties** a nastavit příslušné cesty. Ve zmíněném případě bude cesta k obrázkům a k přijatým souborům :

```
Image_directory = D:\\MATLABServerPages\\webapps
                        \\MATLABServerPages\\Image
```

[3] Java Software Development Kit minimální verze 1.4.2.08 a balíček MSP

Obr. 3.2: MSP – Cesta k obrázkům a přijatým souborům.



Obr. 3.3: MSP – Nastavení cesty k Matlabu.

```
Code_directory = D:\\MATLABServerPages\\webapps
                          \\MATLABServerPages\\Code\\
```

Zvýšenou pozornost věnujte dvojitým lomítkám [4] a absenci lomítek u adresáře na obrázky. Obsah souboru **MSP.properties** viz obr. 3.2

Konfigurace cesty k Matlab COM serveru

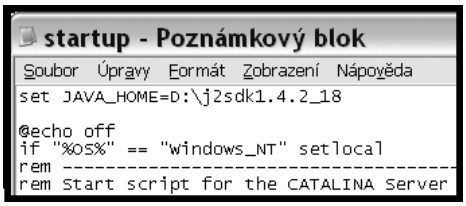
V tomto kroku se nastaví cesta k programu Matlab, na kterém budou prováděny výpočty. Pozdější komunikace bude probíhat přes příkazový řádek Matlabu.

Označením zástupce **MSP – Register MATLAB as COM Server** a **upravením** se musí nastavit cesta k Matlabu viz obr. 3.3. Středem zájmu bude opět první řádek souboru. Pro vzdálený přístup k Matlabu viz informace na [9].

Konfigurace cesty k JSDK

Označením zástupce **MSP – Start up server** a **upravením** se musí nastavit cesta k vývojovému prostředí Javy viz obr. 3.4.

[4] zdvojené zpětné lomítka vyjadřují v jazyce Java znak zpětného lomítka



Obr. 3.4: MSP – Nastavení cesty k JSDK.

Spuštění a ukončení MSP serveru

Po úspěšném nastavení v předchozích krocích nastal čas ke spuštění serveru. Toho lze dosáhnout spuštěním zástupce **MSP – Start up server**. V nově vzniklém okně nyní běží Apache Tomcat Server a je zde možné sledovat všechny jeho úkony a případně i vzniklé chyby za běhu.

Server se ukončí známým klávesovým dvojhmatem pro ukončení[5] programů: **ctrl+c**, ovšem musí být aktivně otevřené okno s běžícím serverem.

Testování vytvořených aplikací

Komunikace se serverem Apache probíhá implicitně na portu **8080**. K prohlížení uvítací stránky je potřeba do prohlížeče napsat adresu i s portem. Jednou z možností je napsat adresu začínající **localhost**, kterou dáme prohlížeči najevo, že se odkazujeme „sami na sebe“. Poté stačí doplnit port, na kterém komunikujeme, a cestu k rodičovskému adresáři našeho souboru. Hledaným souborem je **index.mlsp**. V tomto případě úplná cesta[6] tedy bude:
http://localhost:8080/MATLABServerPages/index.mlsp[7].

3.3 Jednoduchá aplikace

3.3.1 Zvolený program a základní princip

Pro ukázku byl zvolen program ze standardních ukázkových funkcí obsažených v adresáři s MSP.

```
D:\MATLABServerPages\webapps\MATLABServerPages\Pages\
```

Konkrétně jde o stránku **simplerequest**, která zavolá přidruženou funkci **simpleresponse**. Princip celého programu je v jednoduché žádosti ze strany uživatele a jednoduché odpovědi od serveru.

V první fázi uživatel zadá do textového pole číslo a potvrdí tlačítkem **Submit** viz obr. 3.5.

V druhé fázi spustí na serveru s Matlabem příkazová řádka Matlabu a předají se odeslané data. Po vyhodnocení se odešlou zpátky. Nakonec se výsledek zobrazí uživateli v otevřeném okně webového prohlížeče, viz obr. 3.6.

[5] většinou násilné

[6] fyzické mapování na disku vypadá asi takto: **d:/MATLABServerPages/webaps/MSP/**

[7] lze uvádět cestu i bez **index.mlsp**, v konfiguračním souboru webového serveru je nastaveno vyhledávání vhodného souboru – **index.*** a jiný **index** ve složce není

Obr. 3.5: MSP – Ukázka jednoduchého požadavku.

Obr. 3.6: MSP – Ukázka jednoduché odezvy.

3.3.2 Co se skrývá ve zdrojovém kódu

Zdrojový kód jakékoliv stránky využívající možností MSP musí mít příponu **.mlsp** a v hlavičce musí být vložen skript, který je na třetím řádku kódu [8] umístěném pod tímto textem.

Tento zdrojový kód zajišťuje jen zobrazení příslušných formulářových jednotek - textového pole a tlačítka pro odeslání dat.

Simplerequest.mlsp:

```
<html>
<head>
<%@ include file="/Scripts/header.inc" %>
<title>MSP Simple Request Test</title>
</head>
<body>
<form action="simpleresponse.mlsp" method="get">
<input type="text" name="var1"/>
    <input type="submit" name="submit" value="Submit"/>
</form>
</body>
</html>
```

Daleko zajímavější je následující část. Mimo obvyklé HTML značky a skript pro MSP, jsou zde užity příkazy pro komunikaci s Matlabem. Spuštění a ukončení příkazové řádky Matlabu mají za úkol párové tagy `<matlab:Engine>` a `</matlab:Engine>`.

[8] jedná se o prosté vyjádření vložení souboru v programování JSP stránek, soubor je zde uložen v adresáři **Scripts** a nese název **header.inc**

Mezi nimi jsou uvedeny jednotlivé příkazy, obsah a vysvětlení všech příkazů je k nalezení v příloze B.1. V tomto případě se jen jedná o předání čísla poslaného z odpovědního formuláře a uložení do proměnné **a** v Matlabu a následné poslání obsahu této proměnné k uživateli.

Simpleresponse.mlsp:

```
<html>
<head>
<%@ include file="/Scripts/header.inc" %>
<title>MSP Simple Response Test</title>
</head>
<body>
<matlab:Engine>
<matlab:Command cmd="a=${param.var1}"/>
<matlab:WriteData name="a"/>
</matlab:Engine>
</body>
</html>
```

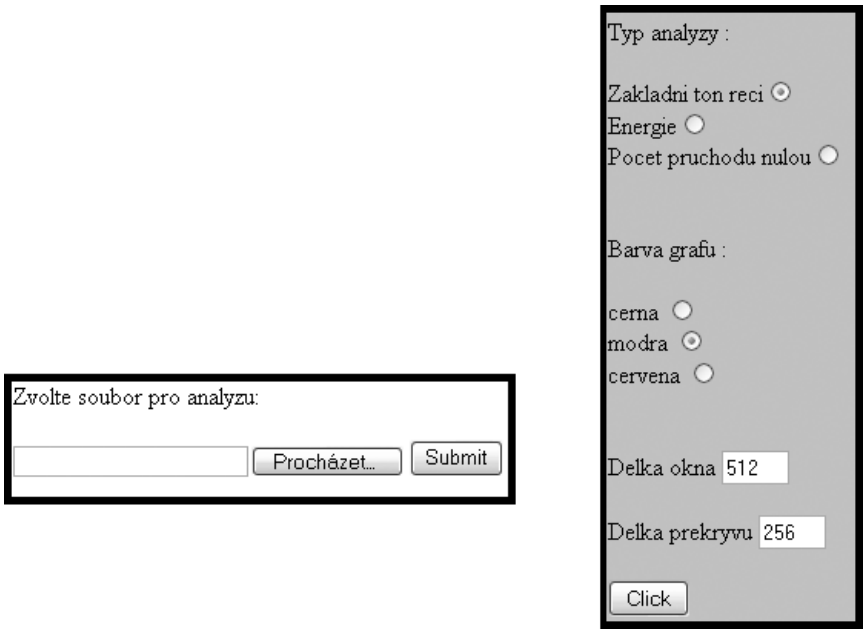
3.4 Aplikace č.1

3.4.1 Úvod

Tato jednoduchá aplikace vznikla jako jednoduchá ukázka schopností MSP projektu. Po konzultaci s garantem této práce byla zvolen program pro jednoduchou analýzu řeči. Cílem této práce není vysvětlení principů takové analýzy, ale program, který tento požadavek provede. Úkolem této implementace je přijmout od uživatele soubor ve formátu **.wav**, poslat jej na analýzu do prostředí Matlab, která je k takovým úkonům vybavena vhodnými funkcemi – v tomto případě M-File(Matlabovský soubor s funkcemi – Matlab file), a zobrazit výsledek na webových stránkách.

3.4.2 Program

Ke zkrácení cesty nutné k přístupu k programu přes webový prohlížeč je použito jednoduché přesměrování, zdrojový kód je umístěn v příloze B.2. K uploadnutí souboru na server je použita poupravená funkce obsažená v ukázkových **JSP** stránkách balíčku MSP s názvem **UploadInput.jsp**. Jedná se o jednoduchou webovou stránku na bázi Javy. Stránka obsahuje nahrávací formulář, pro nahrání požadovaného souboru, a tlačítko **Submit**, které zavolá objekt typu JavaBean **uploadBean**, který se o tento soubor postará a uloží ho na správné místo na disku. Ukázka této stránky viz



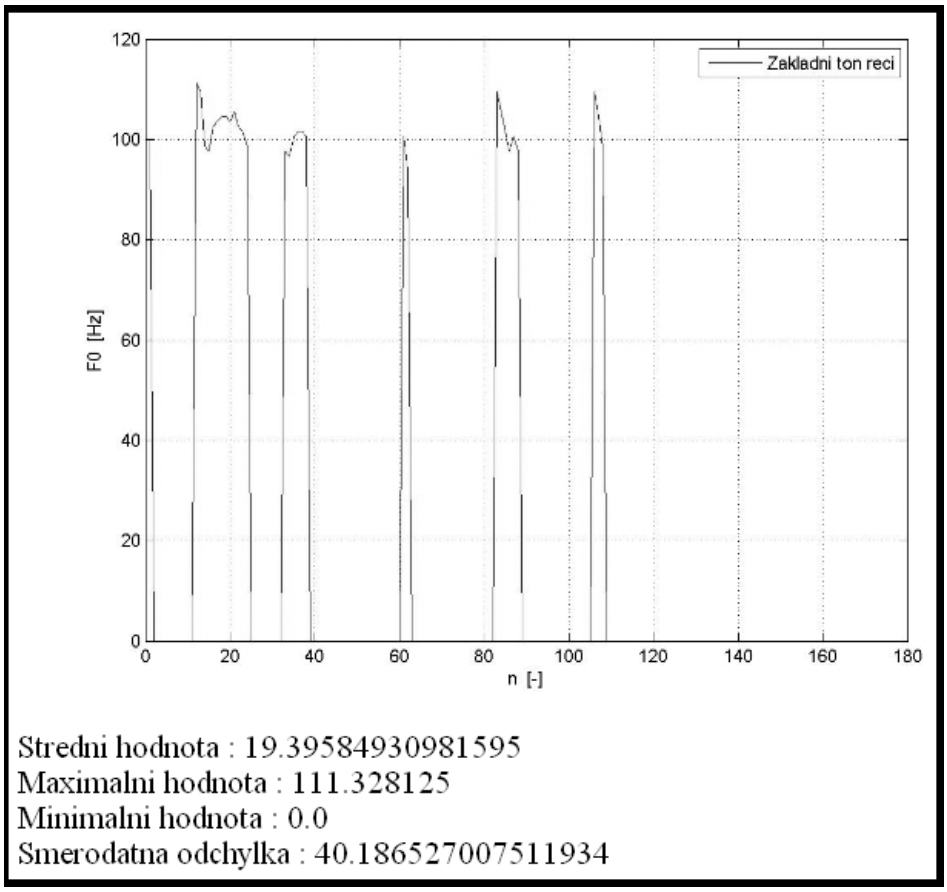
Obr. 3.7: MSP – Program č.1 – Formulář na upload souboru a Formulář s volbou analýz.

obr. 3.7. Soubor je umístěn podle nastavené cesty v souboru **MSP.properties** nebo **MSPRMI.properties**, tedy implicitně do složky **Code** [9]. Obsah těchto souborů se nastavoval v kapitole 3.2.3.

Při správném uploadnutí dojde pomocí návratové hodnoty metody **uploadBeanu** a XML dokumentu k přesměrování na stránku s volbami pro analýzu řeči **vyberAnalyzy.jsp**. Tato stránka je k nahlédnutí na obrázku obr. 3.7. Špatně zadaná data nejsou ošetřena. Po navolení druhu analýzy, barvy grafu a dvou délek se tyto parametry přesměrují na stránku **analyzaAVypis.mlsp** pomocí tlačítka **Click**. Program(stránka) přijme parametry, uloží do proměnných v Matlabu a zpracuje požadavek. Výstup je možné pozorovat na obr. 3.8.

Okomentované zdrojové kódy jsou umístěny v přílohách. Konkrétně stránky pro upload B.3, uploadBean B.4, XML dokument s přesměrováním B.5, stránky s výběrem analýzy B.6, analýza a výpis výsledků B.7 a M-File s analýzou B.8.

[9] úplná cesta v tomto případě: **D:/MATLABServerPages/webaps/MSP/Code/**



Obr. 3.8: MSP – Program č.1 – Výsledek.

4 APLIKACE Č.2

4.1 Úvod

Tato část je věnována vytvořené aplikaci, pracující na bázi klient-server, s možností hromadného zpracování multimediálních souborů i dálkové správy serveru. Vznik této práce koření v zadání bakalářské práce.

Pro korektní fungování celé aplikace se předpokládá server vybaven prostředím Matlab, ve kterém se provádí všechny výpočty a analýzy, Apache Tomcat Web Serverem, který zpracovává JSP(Webové stránky na bázi Javy – Java Server Pages) stránky, ve kterých je soustředěna veškerá komunikace mezi uživatelem – klientem, a serverem, a který zároveň slouží jako webový server – umožňuje zobrazení jejich obsahu ve formě HTML dokumentu všem připojeným klientům. Dále je potřeba JVM(Java Virtuální stroj – Java Virtual Machine), který je nutný pro běh veškerých Java programů, tedy i k chodu JSP stránek. V neposlední řadě je potřeba mít knihovnu **JMatlink.dll** umístěnou v adresáři s operačním systémem [1].

V tomto oddíle se rozvíjí nabyté zkušenosti z předešlé kapitoly v trochu odlišné formě programování. Práce se soustřeďuje spíše do možností aplikace než do možností Matlabu. Dále nevyužívá tolik programů a značek(tagů) z opensource projektu MSP, ale jen jeho základních knihoven ke komunikaci s prostředím Matlab. Nicméně název zůstal stejný – MSP – MATLAB Server Pages.

4.2 Spuštění

Pokud je Vaše PC vybaveno potřebnými programy [2], nahrajte příslušný webový archív(WAR) s aplikací MSP do adresáře **webapps**, který je umístěn v adresářích programu **Apache Tomcat Web Server**. Po spuštění programu **Apache Tomcat Web Server** se archív s aplikací rozbalí a je připraven k spuštění. **Tomcat** se spustí přes **tomcat*.exe** [3], který je v adresáři **bin** tohoto programu. Poté stačí do prohlížeče napsat URL, odkazující se na tento dokument. Pokud jste server nijak nenastavovali, adresa by měla být **http://localhost:8080/MSP**. Po chvíli by se měla objevit stránka s informacemi o projektu, jak je tomu na obr. 4.2. Mezitím program vykonal několik instrukcí, které se provedou vždy po restartu aplikace:

1. uložil do paměti cesty k potřebným souborům a adresářům

[1] např. **c:/windows/**

[2] OS Windows XP, Apache Tomcat Web Server, webový prohlížeč, knihovna JMatlink.dll, prostředí Matlab a nějaký JVM

[3] kde hvězdička značí hlavní verzi programu, například Tomcat 6.0.1.8 bude mít spouštěcí soubor **tomcat6.exe**

2. nastavil soubor **MSP.properties**, který je potřebný pro běh programu **goOnline**, který zprostředkuje komunikaci s Matlabem.
3. nahrál do paměti informace o všech doposud registrovaných uživatelích. V případě, že ještě žádní nebyli vytvořeni, vytvoří uživatelský účet administrátora **superadmin** s nejvyššími právy – **superadmin** a heslem, pro jednoduchost, rovněž **superadmin** a informace uloží do souboru **acc.txt**.
4. nahrál do paměti informace o SMTP(Jednoduchý protokol pro přenos mailů – Simple Mail Transport Protocol) serveru, v případě, že soubor neexistuje, vytvoří jej a uloží do něj IP adresu PC, na kterém program právě běží. Tato skutečnost je vytvořena kvůli předpokladu, že daný server bude vybaven i SMTP serverem [4].
5. zkontroluje, zda-li není nějaký účet určen k smazání [5].
6. načte do paměti všechny analýzy z příslušného adresáře.

4.3 Systematizace souborů v projektu

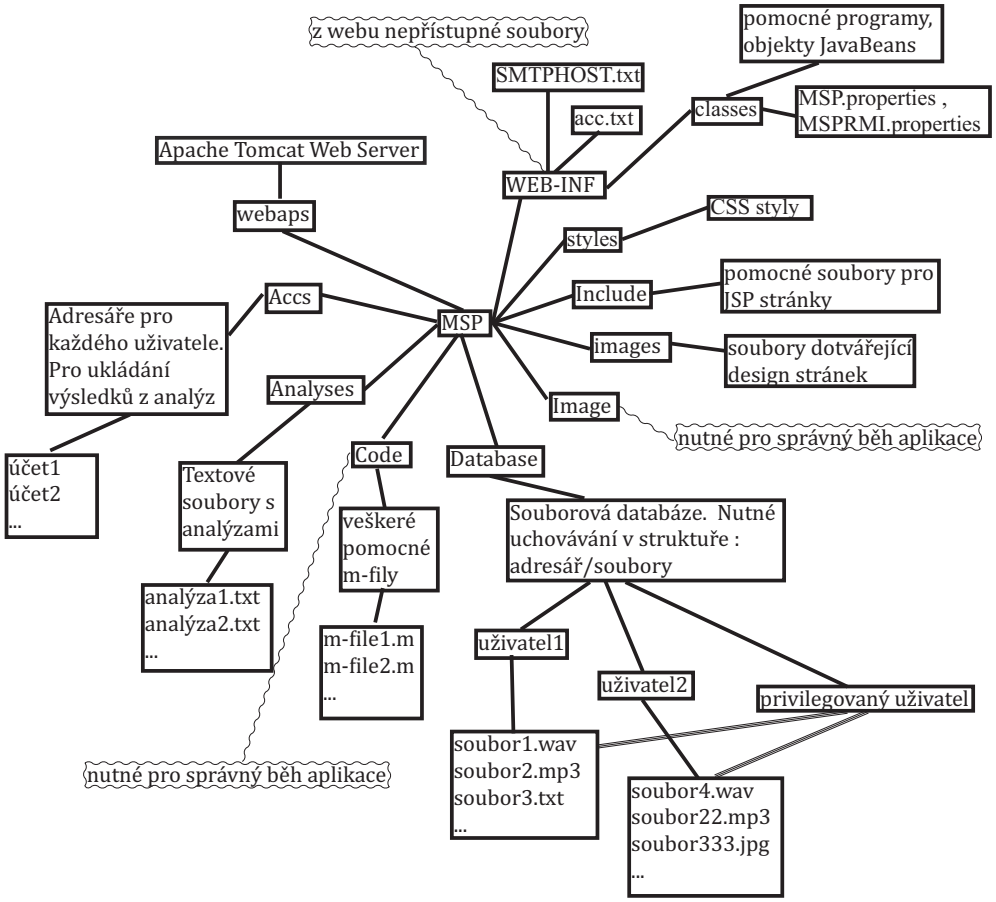
Aplikace je vytvořena především pro operační systém MS Windows a Apache Tomcat Web Server. Pod jinými systémy nemusí fungovat správně, sice je dodržena konvence o oddělování souborů – lomítka jednoduché a zpětné, která si program pro konkrétní systém vybere správně, avšak předpokládá se přesné uspořádání souborů ve složce s instalací Apache Tomcat Web Serveru na systému Windows, tak jak je tomu na obr. 4.1. Jak je vyznačeno na obrázku, pro správný běh aplikace je nutné mít složky **Code** a **Image**. Nutnost jejich existence spočívá v programech, které komunikují s Matlabem. Složka **Image** působí jako odkladiště dočasných souborů – obrázků a druhý adresář se při zahájení každé komunikace označí jako pracovní adresář Matlabu. Mezi další esenciálními složky patří soubory **MSP.properties** a **MSPRMI.properties**, ve kterých jsou namapovány právě předešlé adresáře, tedy adresář **Code** a **Image**. Navíc, oproti původní verze MSP, je v nich ještě 3. adresa – kořenová adresa aplikace [6], pro snazší tvorbu relativních URL(Jednoznačné určení zdroje – Uniform Resource Locator), potažmo odkazů na potřebné soubory. Veškeré výše zmíněné složky i soubory se vytvoří automaticky při startu aplikace[7]. Přehled složek a jejich význam je uveden v tab. 4.1.

[4] SMTP server zajišťuje například program **PostCast Server**, který je volně k stažení viz [13]

[5] tato kontrola probíhá při každé změně práv na úroveň uživatele k vymazání. Je zde jen pro jistotu, že by aplikace nenadále zhavarovala, ještě než by se příslušný soubor s účtem stihl upravit.

[6] kompletní cesta až k ... Apache Tomcat Web Server/webaps/MSP/

[7] program, který cesty doplní ovšem předpokládá výše zmíněnou souborovou strukturu a spuštění v Apache Tomcat Web Serveru



Obr. 4.1: Aplikace č.2 – Systém souborů.

Tab. 4.1: Tabulka s významy složek aplikace č.2.

název složky	význam
MSP	kořenový adresář aplikace
Accs	složka se složkami pro všechny registrované uživatele. Uživatelova složka je vytvořena ihned po první nalogování, dále se její existence kontroluje jen při dalším logování. Každá uživatelova složka obsahuje složku s výsledkami analýz, v té je přehledný soubor složek s datem spuštěné analýzy a uvnitř jsou exportované výsledky – obrázky a data.
Analyses	složka, do které se ukládají postupy analýzy. Každá analýza je ve svém textovém souboru. Postupy se čtou při startu aplikace, nebo při zjištění přidání nové analýzy.
Code	složka k ukládání pomocných m-filů, při každé relaci s Matlabem se využívá jako pracovní adresář. Nutná k správnému běhu(nemusí v ní ani nic být).
Database	složka se souborovou databází. Každý uživatel má svojí vlastní složku pro nahrané soubory. Každý dostatečně privilegovaný uživatel může využívat soubory s ostatních složek. Viditelný obsah pro uživatele se obnovuje při každém požadavku.
Image	složka pro odklad exportovaných obrázků z Matlabu. Nutná pro chod programu. V aplikaci se už nevyužívá.
images	složka pro veškeré obrázkové soubory dotvářející design stránek.
Include	složka s částmi html dokumentu pro přehlednější a méně objemný kód při programování a rozcestníkem, který se pro každého uživatele mění dle jeho aktuálních práv a stavu.
styles	složka s CSS styly.
WEB-INF	standardní složka serveru Apache Tomcat. Z webu v původní konfiguraci nepřístupná. Díky této vlastnosti, se v této aplikaci používá i jako úložiště souboru **acc.txt** , který obsahuje uživatelské data (jména, hesla, email). Dále je zde soubor **SMTPHOST.txt**, ve kterém je obsažena adresa SMTP serveru, který zajišťuje správné doručení registračních emailů, pokud je ovšem tato volba zapnuta. Navíc tato složka slouží standardně jako odkladiště veškerých pomocných Java programů.

Tab. 4.2: Tabulka s významy uživatelských práv aplikace č.2.

název práva	význam
User to be deleted	Uživatel k vymazání.
Watcher	Stav uživatele po příchodu na stránky.
User waiting for confirmation	Uživatel čekající na odsouhlasení registrace od administrátora. Stav uživatele po registraci.
Normal user	Možný stav uživatele. Uživatel má standardně práva k nahrávání souborů, mazání souborů ve vlastní složce a ve složce s výsledkami analýz. Smí spouštět analýzy.
Super user	Stejné pravomoce jako **Normal user**. Navíc může užívat soubory ze složek ostatních uživatelů.
Exclusive user	Stejné pravomoce jako předchozí. Vytvořen jen pro větší rozmanitost pravomocí.
Admin	Pravomoce na promování uživatelů až na úroveň **Exclusive user**. Standardně i pravomoce pro mazání souborů mimo uživatelovu složku, pro vytváření analýz, upravování velikostních limitů nahraných souborů, změna SMTP serveru, změna volby pro posílání nebo neposílání emailů o registraci uživatele, změna minimálních pravomocí pro: výmaz souborů mimo složku a k vytváření analýz.
Super Admin	Nejvyšší pravomoce, platí to co u předchozího práva. Navíc má možnost promovat uživatele až na hodnost **Admin**.

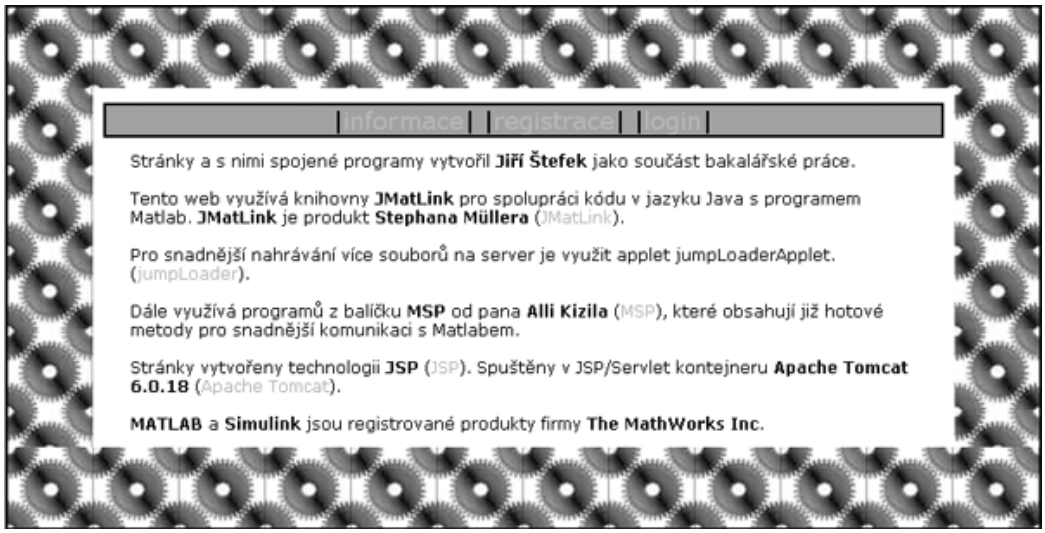
4.4 Uživatelská práva

V aplikaci je rozlišeno 8 druhů práv nebo lépe vystiženo – stavů uživatele. Jednotlivá práva i s komentářem jsou uvedena v tab. 4.2.

4.5 Možnosti

Výčet schopností aplikace:

1. správa uživatelských účtů. Vytvoření i mazání uživatelských účtů, oznamování emailem, změna uživatelských práv, změna hesla.
2. analýzy. Jednoduchá tvorba, úprava i mazání analýz s požadavkem na základy programování v Matlabu. Možnost využití M-Filů. Automatické vytvoření nabídek pro spuštění analýzy. Analýzy více souborů najednou. JavaScripty zajiš-



Obr. 4.2: Aplikace č.2 – Info stránka.

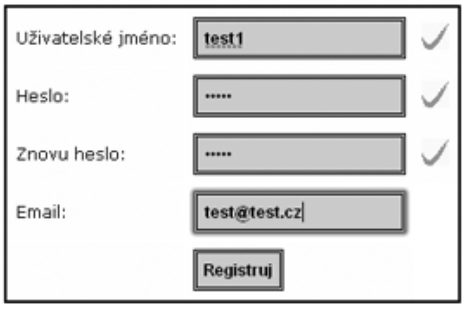
ťující správné zvýraznění použitelných souborů pro danou analýzu. Přehledné uspořádání výsledků analýzy.

3. vzdálená manipulace se soubory na serveru. Nahrávání a mazání souborů v uživatelských složkách a s příslušnými právy i mimo ně. Proces nahrávání prováděn přes Java Applet s možností uploadovaní více souborů naráz a se zobrazením aktuálního stavu přenosu.
4. kódování. Všechny stránky jsou kódovány ve formátu UTF-8. Tento formát zajišťuje správné zobrazení češtiny a správnou manipulaci s českými názvy.

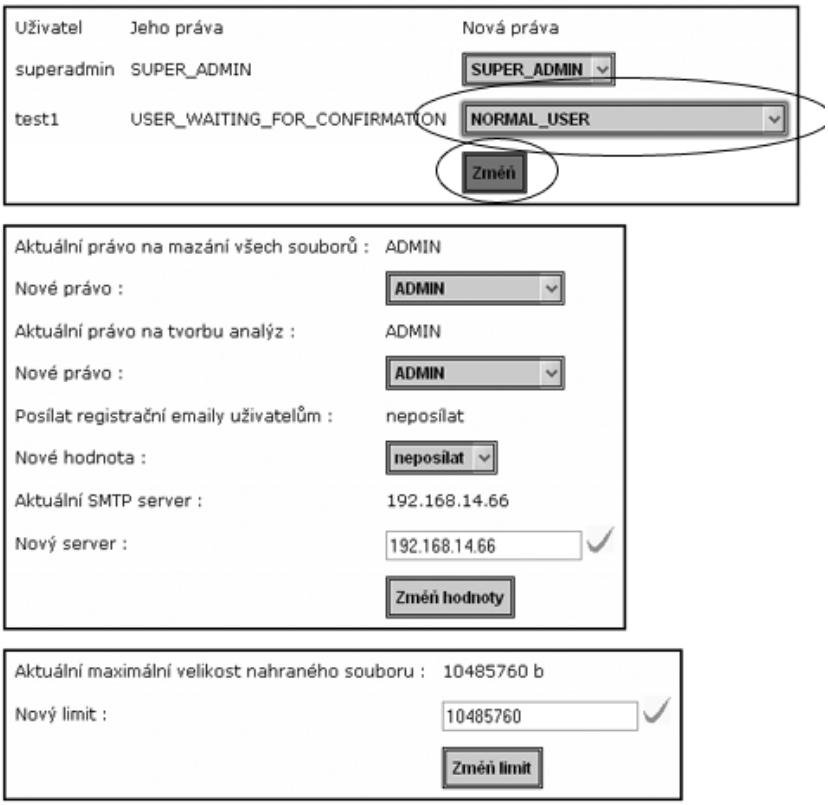
4.6 Ukázka aplikace

Po spuštění aplikace[8] a zadání adresy do prohlížeče nás uvítá informační stránka jako je na obr. 4.2. Pro zpřístupnění veškerých funkcí aplikace je nutná registrace. Z uživatelovi strany se stačí zaregistrovat pomocí formuláře jako je na obr. 4.3, kam se dostane přes odkaz **registrace**. Z obrázku je patrné, že k registrací stačí jen jméno a heslo. Heslo ani jméno nesmí obsahovat jiné znaky než anglické abecedy nebo čísel. Po úspěšné registraci se zobrazí o tomto faktu zpráva. Z administrátorovi strany je potřeba přidělit uživateli práva. Administrátor je kdokoliv s právy většími nebo rovnými statutu **Admin**. Seznam všech možných práv je uveden v tab. 4.2, přitom k přístupu do aplikace stačí práva **Normal user**. Po přihlášení administrátora se práva nastaví přes odkaz **administrace**, nastavení je uvedeno na obr. 4.4. Po úpěšném nastavení práv se nyní může uživatel přihlásit.

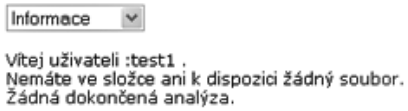
[8] probráno v kapitole 4.2



Obr. 4.3: Aplikace č.2 – Registrace.



Obr. 4.4: Aplikace č.2 – Registrace 3.



Obr. 4.5: Aplikace č.2 – Přihlášený.

Vyplněním přihlašovacích údajů v odkazu **login** se uživatel přesune na stránku viz obr. 4.5. K této stránce se uživatel dostane přes odkaz **účet**. Na první pohled má jen informativní charakter, ovšem po rozbalení roletového menu si uživatel může nastavit nové heslo.

Protože je aplikace koncipována způsobem nahraj soubor – spusť si jeho analýzu, je nutné mít přístup k nějakému souboru. Jestliže vlastník účtu má práva minimálně **super user**, smí spouštět analýzy jiných souborů, než které má ve své složce. V podstatě toto řešení umožňuje využití společné souborové databáze nahrávek všem uživatelům s potřebnými právy. Nahrát soubory lze buď přes applet přístupný přes odkaz **nahrát soubory** nebo za předpokladu, že máte k disku s webovým serverem přímý přístup, přímo do podsložky složky Database [9]. Applet umožňuje mimo jiné i funkci *Drag and Drop*, což určitě přidá na uživatelově pohodlí a taktéž zobrazuje současný stav přenosu souboru. Applet je znázorněn na obr. 4.6, v kroužku nahoře jsou označeny soubory k uploadu a dole je tlačítko k provedení. Po úspěšném nahrání souborů na server lze spouštět analýzy.

Okno s analýzami se otevře přes odkaz **analýzy**. Po odkliknutí se zobrazí tabulka s dostupnými analýzami a taktéž seznam přístupných souborů jako na obr. 4.7. Zvolením analýzy pomocí radiového tlačítka, označením potřebných souborů pomocí zaškrtávacích políček ležících na stejné řádce vedle názvů souborů a spuštěním analýzy pomocí tlačítka **run** se spustí analýza. Po ukončení procesu se v prohlížeči objeví zpráva o provedení a uplynulém čase [10] a o cestě, kde daný výsledek hledat.

Po otevření odkazu **výsledky analýz** se zobrazí přehled všech souborů exportovaných z provedených analýz, tak jak je tomu např na obr. 4.8. Obsah označených souborů je na obr. 4.9 a obr. 4.10. Jak je vidět, obrázky jsou ve formátu **EMF** [11] a exportované proměnné v prostém textovém souboru **txt**. Formát textového souboru je následující: zvolený název proměnné, rozměry výsledku [12] a samotný výsledek.

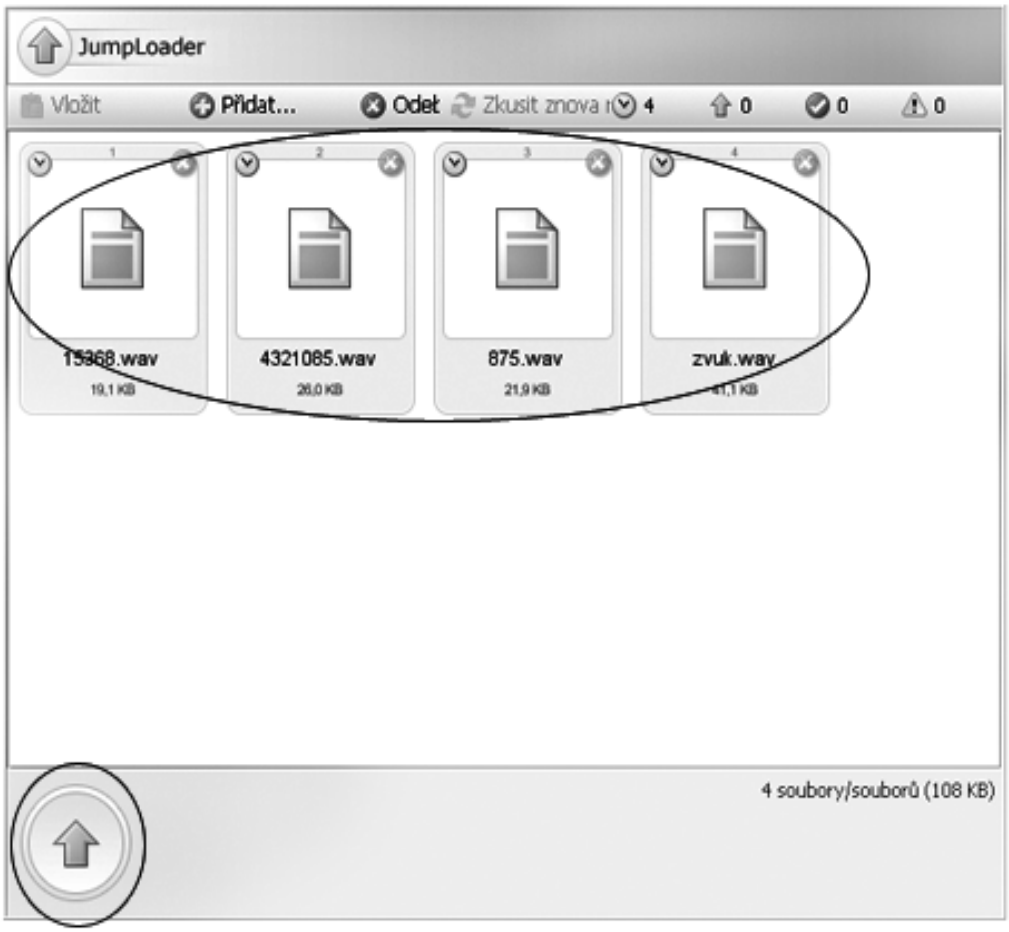
Pro změnu nebo vytvoření analýz je potřeba mít práva pro změnu analýz, standardně nastavené na privilegia účtu minimálně **admin**. Ovšem tato minimální hra-

[9] např. **D:/Apache Tomcat/webaps/MSP/Database/uživatelovoJméno/**, ne však přímo **D:/Apache Tomcat/webaps/MSP/Database/**, program je totiž takto navržen

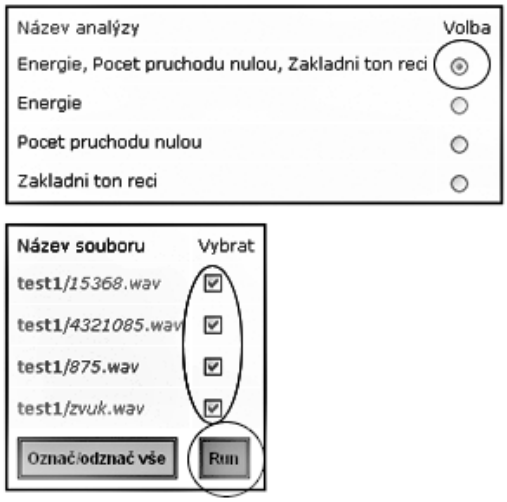
[10] čas, který samotná analýza trvala, nikoliv celkový čas od zadání analýzy

[11] změna formátu je v současném stavu možná jen změnou v kódu souboru **MatlabBean** v metodě **plotData**

[12] zde je uvedeno 1 4, což odpovídá matici 1x4, respektive vektoru o 4 prvcích

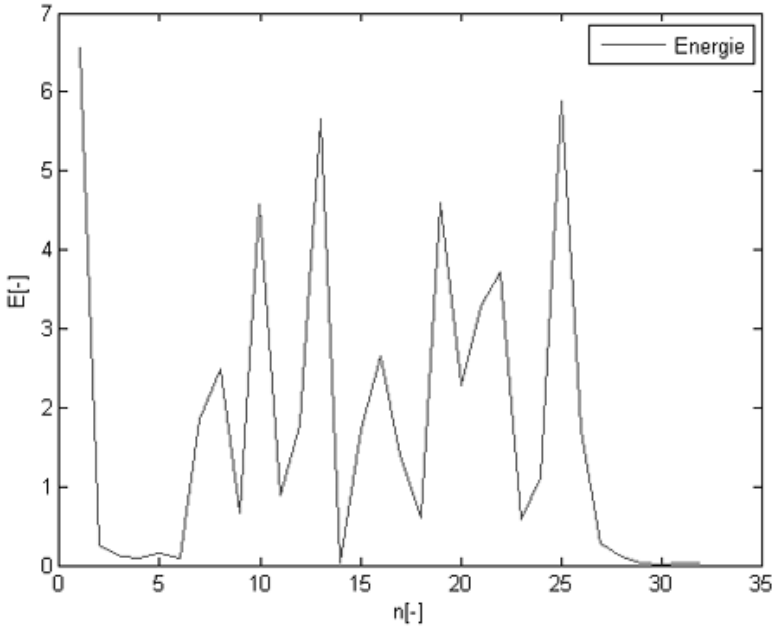


Obr. 4.6: Aplikace č.2 – Nahrávání souborů.



Obr. 4.7: Aplikace č.2 – Výběr analýzy.

Obr. 4.8: Aplikace č.2 – Výsledky analýzy.



Obr. 4.9: Aplikace č.2 – Výsledek analýzy – obrázek.

```
Eparam:Stredni hodnota,Maximalni hodnota,Minimalni hodnota,Smerodatna odchylka
1 4
1.7265644073486328 6.5626220703125 0.01654052734375 1.9473804011201274

F0param:Stredni hodnota,Maximalni hodnota,Minimalni hodnota,Smerodatna odchylka
1 4
54.730224609375 142.578125 0.0 63.433632240974745

ZCRparam:Stredni hodnota,Maximalni hodnota,Minimalni hodnota,Smerodatna odchylka
1 4
131.0625 740.0 0.0 175.38104828280106
```

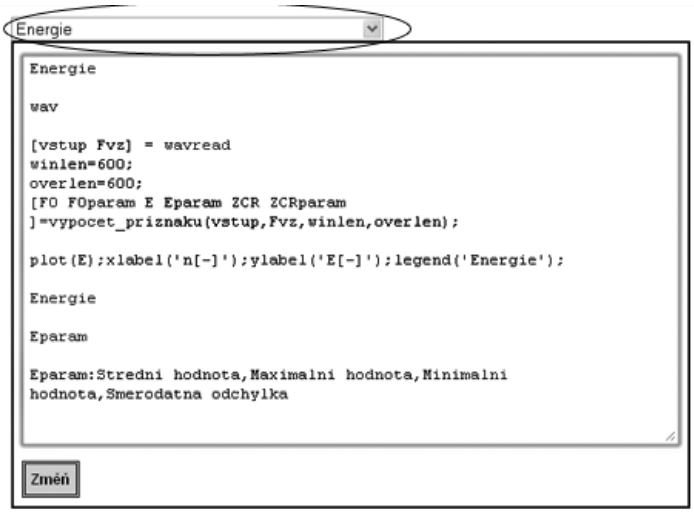
Obr. 4.10: Aplikace č.2 – Výsledek analýzy – text.

nice lze změnit přes odkaz **administrace**. Rozvinutím roletového menu lze volit mezi novou analýzou nebo úpravou již existující. Na obr. 4.11 je vidět úprava analýzy s názvem **Energie**. Analýzy se ukládají v tomto formátu:

1. název analýzy, pokud možno ne český – JMatLink se neumí bohužel vypořádat s UTF-8 kódováním. Název analýzy se využívá k pojmenovávání souborů.
2. volný řádek.
3. možné přípony souborů.
4. volný řádek.
5. příkazy pro Matlab. U příkazu **wavread** se jako vstupní argument automaticky doplní použitý soubor [13].
6. volný řádek.
7. příkazy k vykreslení grafů, které se budou exportovat. Legenda.
8. volný řádek.
9. názvy, pod kterými se budou grafy ukládat.
10. volný řádek.
11. názvy proměnných v Matlabu, které se mají exportovat.
12. volný řádek.
13. názvy pod kterými budou v exportovaných souborech vystupovat.

Další podmínky:

[13] pro nastavení jiných příkazů pracujících se soubory jako vstupními argumenty, je nutná editace zdrojového kódu, přidáním dalšího řetězce do proměnné **fileHandlingCommands** v souboru **MatlabBean**.



Obr. 4.11: Aplikace č.2 – Úprava analýzy.

1. název analýzy je jen jeden.
2. příkazy se píšou pod sebe, každý na svůj řádek. Pokud jde o příkazy k vykreslení grafů, píší se informace ohledně jednoho grafu na jeden řádek. Tak jak je tomu na obrázku 4.11.
3. každý vykreslený graf musí mít svůj název, stejně tak i každá exportovaná proměnná. Nedodržením této konvence se analýza smaže.
4. ke smazání analýzy stačí vymazat celé textové pole a potvrdit změny.

Ukázkový příklad:

1. dumyslna analyza
2. volný řádek
3. wav
4. volný řádek
5. `[promenna1 promenna2]`=readwav
6. volný řádek
7. plot(promenna1);legend('toto je graf 1');xlabel('osa x');ylabel('osa y');
8. volný řádek
9. casovy prubeh 1
10. volný řádek
11. promenna1
12. volný řádek
13. vzorky 1

Hot Spots - Method	Self time	Invocations
Beans.MatlabBean.**printResultToFile** (String[][][], String[], String, String)	52.0 ms	40
Beans.ApplicationBean.**refreshFileDatabaseList** ()	33.0 ms	20
Beans.MatlabBean.**evaluateCommand** (String)	16.5 ms	520
org.apache.jsp.analyse_jsp.**_jspService** (javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServletResponse)	13.2 ms	15
Beans.MatlabBean.**abstractAnalyse** (String, String[], String[], String[], String[], java.io.File[], String)	8.34 ms	10
Beans.MatlabBean.**analyseResultsMakepath** (String)	7.4 ms	10
Beans.MatlabBean.**open** ()	6.81 ms	10
Beans.MatlabBean.**plotData** (String, String, String, String)	5.35 ms	120
Beans.MatlabBean.<**init**> ()	5.19 ms	15
Beans.LoggedUserBean.**createUsersDirs** ()	3.57 ms	5
Beans.ApplicationBean.**softRefreshAnalyses** ()	3.30 ms	15
Beans.MatlabBean.**getArray** (String)	2.83 ms	120
Beans.LoggedUserBean.**changeToWWWSeparators** (String)	2.0 ms	60
org.apache.jsp.login_jsp.**_jspService** (javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServletResponse)	1.73 ms	8
org.apache.jsp.loggedIn_jsp.**_jspService** (javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServletResponse)	0.852 ms	5
Beans.MatlabBean.**isCommandHandlingWithFiles** (String)	0.835 ms	160
Beans.MatlabBean.**close** ()	0.579 ms	10
Beans.LoggedUserBean.**makeReferencesToFiles** (java.io.File[])	0.423 ms	15
Beans.LoggedUserBean.**refreshAnalyseResultsFilesList** ()	0.330 ms	5
Beans.ApplicationBean.**logInUser** (String)	0.195 ms	5
Beans.ApplicationBean.**getAnalysePossibleExtensions** (String)	0.175 ms	85
Beans.LoggedUserBean.**logUserAndCreateDirs** (String, int, int, String, String, String)	0.127 ms	5
Beans.ApplicationBean.**getFileDatabaseList** (String, int)	0.090 ms	20
Beans.ApplicationBean.**getAnalysesNames** ()	0.073 ms	15
org.apache.jsp.analyse_jsp.**getDependants** ()	0.045 ms	10
Beans.ApplicationBean.**getIndexOfAcc** (String)	0.029 ms	15
Beans.ApplicationBean.**getAnalyse** (String)	0.018 ms	10
Beans.LoggedUserBean.**getAnalyseResultsFilesList** ()	0.012 ms	5
Beans.ApplicationBean.**getUserPass** (String)	0.011 ms	5
org.apache.jsp.loggedIn_jsp.**getDependants** ()	0.008 ms	5
org.apache.jsp.login_jsp.**getDependants** ()	0.007 ms	5

Obr. 4.12: Aplikace č.2 – Zátěžový test CPU.

4.7 Testování

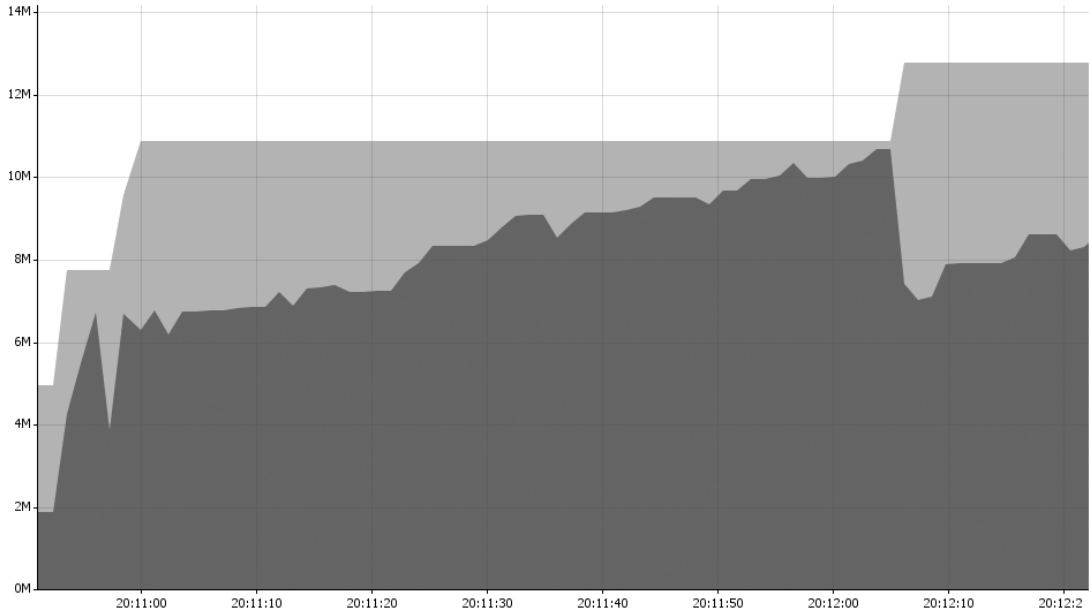
Celá aplikace byla vytvořena v prostředí NetBeans IDE 6.5 (viz [12]). Spouštěna a testována v OS Windows XP SP3 32bit na PC sestavě: procesor AMD Dual Core 4800+ pracující na taktu 2,500 GHz, operační paměť 2x 2GB 667 MHz a disk SATA2 s 16 MB Cache a 7200 RPM. JVM z JDK 1.7. Testováno v prohlížečích Apple Safari 4, Google Chrome 2.0.181.1, Internet Explorer 8.0.6001 a Mozilla Firefox 3.0.7. JSP kontejner Apache Tomcat 6.0.18. Testy vytížení pamětí a procesoru vytvořeny pomocí profilování v NetBeans IDE 6.5.

Výsledek paměťového zátěžového testu pro 5 uživatelů je na obr. 4.12 a 4.13. První obrázek znázorňuje naplnění paměti různými druhy datových typů seřazených podle jejich četnosti. Druhý obrázek ukazuje využití paměti. Světlejší křivka sleduje velikost paměti typu hromada, tmavší sleduje využitou paměť typu hromada.

Výsledek procesorového zátěžového testu pro 5 uživatelů je na obr. 4.14. Obrázek znázorňuje využití metod seřazených podle časové náročnosti. První sloupec obsahuje názvy volaných metod, druhý celkový čas strávený výpočtem metody, třetí je celkový počet volaných metod. Takže průměrná doba výpočtu této metody se dá určit jako podíl celkového času stráveného nad metodou a celkovým počtem volané metody.

Class Name - Live Allocated Objects	Live Bytes ▼	Live Bytes	Live Objects	Allocated Objects	Avg. Age	Generations
char[]		473 264 B (31,5%)	3 360 (17,6%)	57 352	43.8	81
byte[]		231 128 B (15,4%)	118 (0,6%)	1 413	32.8	46
int[]		132 184 B (8,8%)	257 (1,3%)	953	42.1	47
java.lang.**Object[]**		124 424 B (8,3%)	874 (4,6%)	7 977	19.1	55
java.util.**HashMap$Entry[]**		69 712 B (4,6%)	545 (2,9%)	1 692	30.5	74
java.lang.**String**		67 848 B (4,5%)	2 827 (14,8%)	33 747	47.5	77
java.lang.reflect.**Method**		48 960 B (3,3%)	612 (3,2%)	1 686	75.6	19
java.util.**HashMap$Entry**		31 608 B (2,1%)	1 317 (6,9%)	3 107	49.6	77
java.util.**HashMap**		19 240 B (1,3%)	481 (2,5%)	1 527	25.7	69
org.netbeans.modules.schema2beans.**AttrProp**		15 848 B (1,1%)	283 (1,5%)	641	7.7	15
java.lang.**String[]**		13 568 B (0,9%)	182 (1%)	1 724	27.5	42
java.util.**ArrayList**		13 464 B (0,9%)	561 (2,9%)	4 927	15.4	37
com.sun.org.apache.xerces.internal.xni.**QName**		10 224 B (0,7%)	426 (2,2%)	1 416	32.5	17
com.sun.org.apache.xerces.internal.dom.**AttrImpl**		10 016 B (0,7%)	313 (1,6%)	698	7.3	15
com.sun.org.apache.xerces.internal.dom.**ElementImpl**		6 912 B (0,5%)	144 (0,8%)	317	7.3	15
org.netbeans.modules.schema2beans.**BeanProp**		6 528 B (0,4%)	102 (0,5%)	223	7.5	15
java.lang.**Class[]**		6 408 B (0,4%)	370 (1,9%)	12 252	72.0	33
org.apache.tomcat.util.modeler.**AttributeInfo**		6 280 B (0,4%)	157 (0,8%)	157	69.8	20
com.sun.org.apache.xerces.internal.dom.**TextImpl**		5 856 B (0,4%)	183 (1%)	401	5.2	12
java.util.**LinkedHashMap$Entry**		5 632 B (0,4%)	176 (0,9%)	340	24.9	14
org.apache.tomcat.util.buf.**MessageBytes**		4 928 B (0,3%)	88 (0,5%)	88	25.6	14
org.netbeans.modules.schema2beans.**DOMBinding$BeanProperty**		4 768 B (0,3%)	149 (0,8%)	322	7.3	15
org.netbeans.modules.schema2beans.**DOMBinding**		4 736 B (0,3%)	148 (0,8%)	330	7.4	15
java.util.**Hashtable$Entry**		4 728 B (0,3%)	197 (1%)	561	65.5	40
java.util.**Vector**		4 632 B (0,3%)	193 (1%)	503	19.5	38
org.netbeans.modules.web.monitor.data.**Param**		4 512 B (0,3%)	94 (0,5%)	200	7.2	15
org.apache.tomcat.util.buf.**ByteChunk**		4 400 B (0,3%)	110 (0,6%)	110	26.1	17
java.util.concurrent.**ConcurrentHashMap$HashEntry[]**		3 944 B (0,3%)	107 (0,6%)	120	51.9	15
com.sun.org.apache.xerces.internal.xni.**QName[]**		3 776 B (0,3%)	28 (0,1%)	87	32.9	13
java.util.**Hashtable$Entry[]**		3 744 B (0,2%)	68 (0,4%)	219	44.8	34

Obr. 4.13: Aplikace č.2 – Zátěžový test paměti.



Obr. 4.14: Aplikace č.2 – Zátěžový test paměti 2.

5 ZÁVĚR

Prvotním cílem této práce je popsat možnosti prostředí Matlab v ohledu komunikace s jinými programy a programovacími jazyky. Z programů je zde zveřejněn průběh komunikace s programem EXCEL. Z programovacích jazyků jsou zde ukázány spolupráce s jazyky Java a podrobněji jazyk C.

V další kapitole jsou probrány možnosti komunikace s periferiemy. Je zde vysvětlena komunikace přes rozhraní RS-232 a pomocí protokolu pro přenos dat FTP.

Následující oddíl ukazuje možnosti a přednosti opensource projektu MSP a jeho možné využití je ukázáno na aplikaci pro jednoduchou analýzu řeči. Ke všem výše uvedeným možnostem je uveden příklad buďto přímo v textu, nebo v přílohách.

Poslední kapitola odkrývá výsledek práce stavějící na požadavcích bakalářské práce. Cílem mělo být vytvoření aplikace pro hromadné zpracování multimediálních souborů, aplikace měla poskytovat uživateli možnost dálkově spravovat server a využívat poskytované funkce pro zpracování souboru, dále měla aplikace být schopna exportovat výsledky zpracování do souborů různého formátu. Z těchto požadavků jsou snad všechny splněny. Navíc je tu možnost webové prezentace výsledků a s tím, z bezpečnostních důvodů, i vytváření a spravování uživatelských účtů. Na serveru lze dálkově manipulovat se soubory a tvořit analýzy. Na druhé straně, práce si standardně umí poradit jen se soubory ve formátu **wav**, ke kterým je ovšem Matlab jako jediný uzpůsoben bez použití nádstavbových knihoven a funkcí. Programy nejsou napsány nejčistší a nejsprávnější formou a určitě by šly upravit ve směru univerzálnosti, kdyby byly použity všechny možnosti vyššího objektového programování. Taktéž by práce šla rozvinout například pro zpracování jiných zvukových formátů a graficky i funkčně ji jinak vylepšit. Celá aplikace i se zdrojovými kódy a potřebnými programy je umístěna v příloze na CD.

LITERATURA

[1] THE APACHE SOFTWARE FOUNDATION. *Apache Tomcat* [počítačový program]. 2009, [cit. 2009-20-05]. Dostupné z URL: <http://tomcat.apache.org/ >. Program pro kompilaci a spouštění JSP stránek.

[2] HEROUT, P. *Java – bohatství knihoven.* Druhé upravené a rozšířené vydání. ISBN 80-7232-288-5.

[3] BURD, B. *JavaServer Pages: Podrobný průvodce.* První vydání, Computer Press 2003. ISBN 80-7226-804-X.

[4] SUN MICROSYSTEMS, Inc. *JavaServer Pages* [počítačový program]. Ver. 2.1. 3.2.2009 [cit. 2009-20-05]. Dostupné z URL: <http://java.sun.com/products/jsp/ >. Dynamické webové stránky v Javě.

[5] TOFT, I. E. *JLab - Connecting Java to Matlab* [počítačový program]. [Velká Británie], 6.1.2005 [cit. 2009-20-05]. Dostupné z URL: <http://www2.cmp.uea.ac.uk/ it/jlab/index.html >. Knihovna pro propojení prostředí Matlab a Java programů.

[6] NAVIA, Jacob. *lcc-win32: A Compiler system for windows by Jacob Navia* [počítačový program]. Ver. 2009-05-17. 17.5.2009 [cit. 2009-20-05]. Dostupné z URL: <http://www.cs.virginia.edu/ lcc-win32/ >. Kompilátor programů v jazyce C.

[7] HERINGOVÁ, B., HORA, P. *Matlab Díl I. – Práce s programem* [online]. 1995 [cit. 2009-20-05]. Dostupné z URL: <http://www.cdm.cas.cz/czech/hora/vyuka/mvs/tutorial.pdf >.

[8] THE MATHWORKS, Inc. *Matlab* [počítačový program]. Verze 7.1. Program pro vědecko-technické výpočty.

[9] KIZIL, A. *Matlab Server Pages* [počítačový program]. [Turecko], 2007 [cit. 2009-20-05]. Dostupné z URL: <http://msp.sourceforge.net/ >.

[10] ZAPLATÍLEK, K., DOŇAR, B. *Matlab – tvorba uživatelských aplikací.* Nakladatelství BEN, 2005.

[11] ZAPLATÍLEK, K., DOŇAR, B. *Matlab – začínáme se signály.* Nakladatelství BEN, 2007.

[12] SUN MICROSYSTEMS, Inc. *NetBeans IDE* [počítačový program]. Ver. 200811100001. Dostupné z URL: <http://www.netbeans.org/ >. Prostředí pro snadnější vývoj programů, mimo jiné i programů v jazyce Java.

[13] GATE COMM SOFTWARE. *PostCast Server Free Edition* [počítačový program]. Ver. 2.6.0. 28.10.2003 [cit. 2009-20-05]. Dostupné z URL: <http://www.postcastserver.com/download/release.aspx?p=3 >. Program zajišťující mimo jiné i SMTP server.

[14] SUN MICROSYSTEMS, Inc. *Java Standard Edition Development Kit* [počítačový program]. Ver. 1.7. 14.5.2009 [cit. 2009-20-05] Dostupné z URL: <http://download.java.net/jdk7/binaries/ >. Prostředí pro vývoj Java programů.

[15] MÜLLER, Stefan. *Stefan Müller's JMatLink* [počítačový program]. Ver. 1.3.0. [SRN], 26.12.2005 [cit. 2009-20-05] Dostupné z URL: <http://jmatlink.sourceforge.net/ >. Knihovna pro propojení prostředí Matlab a Java programů.

[16] HEROUT, P. *Učebnice jazyka Java*. Třetí rozšířené vydání. ISBN 978-80-7232-355-5.

[17] KARBAN, P. *Výpočty a simulace v programech Matlab a Simulink*. Computer Press 2006.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

DDE Dynamická výměna dat – Dynamic data exchange

DNS Systém doménových jmen – Domain name system

FFT Rychlá Fourierova Transformace – Fast Fourier Transformation

FTP Protokol pro přenos dat – File transport protocol

IP Internetový protokol – Internet protocol

JRE Prostředí pro spouštění Java programů – Java Runtime Enviroment

JSDK Prostředí pro vývoj Java programů – Java Software Development Kit

JSP Webové stránky na bázi Javy – Java Server Pages

JVM Java Virtuální stroj – Java Virtual Machine

MEX Spustitelný v prostředí Matlab – Matlab EXecutable

M-File Matlabovský soubor s funkcemi – Matlab file

MSP Stránky Matlab-Serveru – Matlab Server Pages

RS-232 Doporučené standardizované rozhraní 232 – Recommended standard number 232

SMTP Jednoduchý protokol pro přenos mailů – Simple Mail Transport Protocol

URL Jednoznačné určení zdroje – Uniform Resource Locator

WAR Webový archív – Web ARchive

SEZNAM PŘÍLOH

A První příloha **51**
A.1 Funkce v jazyku C a jeho implementace pro Matlab 51
A.2 Volání Matlabu z jazyka C 53

B Druhá příloha **55**
B.1 MSP příkazy 55
B.2 Program - Přesměrování 56
B.3 Program - Upload Souboru 56
B.4 Program - UploadBean 57
B.5 Program - Přesměrování pomocí dokumentu XML 58
B.6 Program - Výběr analýzy 59
B.7 Program - Výpis výsledků 61
B.8 Program - MFile s analýzou 63

C Třetí příloha **66**
C.1 Obsah CD 66

A PRVNÍ PŘÍLOHA

A.1 Funkce v jazyku C a jeho implementace pro Matlab

```
#include "mex.h" //potrebna knihovna
/**
    Funkce pro secteni dvou vektoru
    @param x - 1.vstupni vektor
    @param y - 2.vstupni vektor
    @param z - vektor s vysledkem
    @param n - shodny rozmer obou vektoru
*/
void sectiVektory(double *vekt1, double *vekt2,
                      double *vysledek, mwSize n)
{
//namisto datoveho typu int se pouziva mwSize,
    mwSize i;
    for (i=0; i<n; i++)
    {
        vysledek[i] = vekt1[i] + vekt2[i];
    }
}
/**
  Bránová funkce
  prava strana = vstupni vektory funkce sectiVektory
  leva strana = vystupni vektor funkce sectiVektory
  @param nlhs - pocet vstupnich argumentu funkce
                pocet prvku leve strany rovnice v Matlabu
  @param nrhs - pocet vystupnich argumentu funkce
                pocet prvku prave strany rovnice v Matlabu
  @param plhs[] - pole s argumenty leve strany
  @param prhs[] - pole s argumenty prave strany
*/
void mexFunction( int nlhs, mxArray *plhs[],
                int nrhs, const mxArray *prhs[])
{
    double *vektor1=NULL;
```

```
    double *vektor2=NULL;
    double *vystupniVektor=NULL;
    mwSize rozmer1,rozmer2; //rozmery vektoru
    //osetreni chybovych stavu
    if(nrhs!=2)
    {
      mexErrMsgIdAndTxt("MyToolbox:sectiVektory:nrhs",
                        "Prosim, zadejte 2 vstupy.");
    }
    if(nlhs!=1)
    {
      mexErrMsgIdAndTxt("MyToolbox:sectiVektory:nlhs",
                        "Prosim, zadejte 1 vystup.");
    }
    rozmer1 = mxGetN(prhs[0]);
    rozmer2 = mxGetN(prhs[1]);
    if(rozmer1!=rozmer2)
    {
      mexErrMsgIdAndTxt("MyToolbox:sectiVektory:prhs",
                        "Zadany nestejne vektory.");
    }
    //"vektory naplnime cisly"
    vektor1 = mxGetPr(prhs[0]);
    vektor2 = mxGetPr(prhs[1]);
    //vytvorime si novy vektor pro vystup
    //rozmer bude roven rozmeru vektoru1,
    //cisla typu mxReal = realne cisla
    plhs[0] = mxCreateDoubleMatrix(1,rozmer1,mxREAL);
    //do vystupnihoVektoru ulozime ukazatel na vystupni
    //promennou v Matlabu
    vystupniVektor = mxGetPr(plhs[0]);
    //volani funkce pro secteni vektoru
    sectiVektory(vektor1,vektor2,vystupniVektor,rozmer1);
}
```

A.2 Volání Matlabu z jazyka C

```
/**
 *  Ukazkovy program pro volání Matlabu z
 *  jazyka C.
 *  Soubor : volaniZC.c
 */

/**
 *  Knihovny
 */
#include <windows.h> // pro vytvoření okénka
#include <stdlib.h>
#include <stdio.h>
#include <string.h> //práce s řetězci
#include "engine.h" // pro komunikaci s Matlabem

#define BUFSIZE 256 //velikost bufferu

//hlavni program
int PASCAL WinMain (HINSTANCE hInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR     lpszCmdLine,
                    int       nCmdShow)
{
Engine *ep; //ukazatel pro přístup k Matlabu
char buffer[BUFSIZE+1];
/**
*  Spuštění Matlabu pro výpočet, s ošetřením chyb
*  při spouštění. Při chybě vykreslí okénko s
*  hláškou "Nelze spustit Matlab"
*/
if (!(ep = engOpen(NULL))){
 MessageBox ((HWND)NULL, (LPSTR)"Nelze spustit Matlab",
           (LPSTR) "volaniZC.c", MB_OK);
exit(-1);
}
//vloží příkaz uvedený v uvozovkách na výpočet do Matlabu
  engEvalString(ep, "x = -80:1:80;");
```

```
engEvalString(ep, "y = -5*x.^3 +3*x.^2 +x;");

//vykreslení grafu s popisem
engEvalString(ep, "plot(x,y);");
engEvalString(ep, "title('Jednoducha ukazka vykresleni
                          kubicke rovnice');");
engEvalString(ep, "xlabel('x');");
engEvalString(ep, "ylabel('y');");

//vložení nulového znaku do bufferu na konec.
buffer[BUFSIZE] = '\0';
//buffer bude zachytávat výstup Matlabu
engOutputBuffer(ep, buffer, BUFSIZE);
// příkaz pro výpis proměnných a jejich rozměrů a typu
engEvalString(ep, "whos");
//Zobrazení proměnné v okénku s tlačítkem "OK".
MessageBox ((HWND)NULL, (LPSTR)buffer,
             (LPSTR) "Proměnné v Matlabu", MB_OK);
engClose(ep); //uzavře Matlab
return(0);
}
```

B DRUHÁ PŘÍLOHA

B.1 MSP příkazy

Výčet tagů pro MSP :

1. Engine: Spustí MATLAB pro výpočet(pro jednoho uživatele).
2. SessionStart: Inicializace „sezení“(session)
3. SessionGet: Vrátí identifikační číslo „sezení“(session).
4. SesisonEnd: Uzavře „sezení“(session).
5. Command: Provede příkaz v MATLABu.
6. Clean: Vymaže všechny .JPG a .BMP obrázky v adresáři na obrázky(Image) a smaže .m soubory v adresáři Code.
7. Debug: Fce pro debugování syntaxe MATLAB příkazů.
8. EvalMDL: Spustí Simulinkový model .
9. EvalMFile: Spustí M-soubor.
10. EvalMLSF: Spustí soubory MSP, které, mohou obsahovat více příkazů pro MATLAB.
11. GetArray: Vrátí dvourozměrné pole z MATLABu.
12. GetVector: Vrátí vektor z MATLABu.
13. GetScalar: Vrátí skalár z MATLABu.
14. GetCharArray: Vrátí řetězec znaků z MATLABu.
15. GetParam: Získá vstupní parametry z HTTP formuláře.
16. PlotData: Vykreslí graf v MATLABu a vrátí obrázek.
17. PlotModel: Vykreslí Simulink model a vrátí obrázek.
18. PlotSim: Vykreslí výsledek ze Simulinku a vrátí obrázek.
19. Thumbnail: Vytvoří miniaturní obrázky grafů z MATLABu, vhodnější pro přenos, a vrátí je.
20. PutArray: Vloží dvourozměrné pole do MATLABu.
21. PutVector: Vloží vektor do MATLABu.
22. PutScalar: Vloží skalár do MATLABu.
23. WriteData: Vrátí proměnnou z MATLAB.

B.2 Program - Přesměrování

Obsah souboru **index.mlsp** :

```
<html>
<head>
//hlavicka matlab server pages mlsp
<%@ include file="/Scripts/header.inc" %>
<title>MATLAB Server Pages</title>
</head>
<body bgcolor="white">
//tentor prikaz zajisti presmerovani
<c:redirect url="/Pages/UploadInput.mlspx"/>
</body>
</html>
```

B.3 Program - Upload Souboru

Obsah souboru **UploadInput.jsp** :

```
<%@ include file="/Scripts/header.inc" %>
<f:view>
   <h:outputText value="Zvolte soubor pro analyzu:"/>
    //vlozeni formulare pro uploadnuti souboru
    <h:form id="UploadForm" enctype="multipart/form-data" >
    <h:messages globalOnly="true" styleClass="message"/>
        //s uploadnutym souborem bude pracovat JavaBean uploadBean
        <ext:inputFileUpload id="matlabFileId"
            value="#{uploadBean.matlabFile}"
            storage="file"
            required="true"/>
    <h:message for="matlabFileId"/>
    //vlozeni tlacitka a nastaveni odezvy
      <h:commandButton value="Submit" action="#{uploadBean.pass}"/>
    </h:form>
</f:view>
```

B.4 Program - UploadBean

Obsah souboru **uploadBean.java** :

```
package matlabBeans; //nazev balicku
//knihovny
import java.io.*;
import java.util.ResourceBundle;
import org.apache.myfaces.custom.fileupload.UploadedFile;
public class uploadBean {
    //vytvoreni nove promenne typu UploadedFile
   public UploadedFile matlabFile;
    //funkce set a get
   public UploadedFile getMatlabFile() {
      return matlabFile;
   }
   public void setMatlabFile(UploadedFile matlabFile) {
      this.matlabFile = matlabFile;
   }
   public String pass() {
      try{
    //promenna pro ulozeni cesty
    //nacte ze souboru *.properties cestu k potrebnym adresarum
          ResourceBundle rb;
          try{
    //nacte cestu z MSP.propeties
              rb = ResourceBundle.getBundle("MSP");
          }
          catch(Exception ex){
    //nacte cestu z MSPRMI.propeties
              rb = ResourceBundle.getBundle("MSPRMI");
          }
    //cesta k adresari na soubory a M-Fily
          String Code_Dir = rb.getString("Code_directory");
          String fullpath = Code_Dir + "\\" +"soubor.wav";
          FileOutputStream fo = new FileOutputStream(fullpath);
    //nahraje soubor do adresare na soubory a M-Fily
    //zmeni mu nazev na soubor.wav
          fo.write(matlabFile.getBytes());
    //navratova hodnota pro .xml
```

```
    //v tomto pripade dojde k presmerovani
          return "PASS";
      }
      catch (IOException e){
          e.printStackTrace();
          return "FAIL";
      }
   }
}
```

B.5 Program - Přesměrování pomocí dokumentu XML

Obsah souboru **my-config.xml** :

```
<?xml version="1.0"?>
<!DOCTYPE faces-config PUBLIC
  "-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.0//EN"
  "http://java.sun.com/dtd/web-facesconfig_1_0.dtd" >
<faces-config>
<!-- managed beans of validateBean -->
<managed-bean>
<managed-bean-name>validateBean</managed-bean-name>
<managed-bean-class>matlabBeans.validateBean</managed-bean-class>
<managed-bean-scope>request</managed-bean-scope>
</managed-bean>
<!-- managed beans of uploadBean -->
<managed-bean>
    <managed-bean-name>uploadBean</managed-bean-name>
    <managed-bean-class>matlabBeans.uploadBean</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
  </managed-bean>
<!-- navigation rules ValidateInput.jsp -->
<navigation-rule>
<from-view-id>/Pages/ValidateInput.jsp</from-view-id>
<navigation-case>
<from-outcome>PASS</from-outcome>
<to-view-id>/Pages/ValidateOutput.jsp</to-view-id>
</navigation-case>
```

```
</navigation-rule>
<!--!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
navigacni pravidlo pro JSP UploadInput.jsp
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-->
    <navigation-rule>
        <from-view-id>/Pages/UploadInput.jsp</from-view-id>
        <navigation-case>
           <!--
           podle navratove hodnoty metody pass
           tridy uploadBean dojde k presmerovani
           v pripade navratove hodnoty "PASS" -->
           <from-outcome>PASS</from-outcome>
           <!--presmerovani na adresu vyberAnalyzy.jsp-->
           <to-view-id>/Pages/vyberAnalyzy.jsp</to-view-id>
        </navigation-case>
    </navigation-rule>
</faces-config>
<!--!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-->
```

B.6 Program - Výběr analýzy

Obsah souboru **vyberAnalyzy.jsp** :

```
<HTML>
<HEAD>
<TITLE>Volby</TITLE>
<body bgcolor="97CCF7">
//javascipt pro zpracovani vstupu
<SCRIPT LANGUAGE="JavaScript">
function vyhodnot (form)
{
//zjisti ktere z radio tlacitek s jmenem "rad" je zaskrtnuto
//jeho cislo ulozi do promenne Count2
  for (Count1 = 0; Count1 < 3; Count1++)
  {
    if (form.rad[Count1].checked)
      break;
  }
/*
```

```
  zjisti ktere z radio tlacitek s jmenem "rad2" je zaskrtnuto
  jeho cislo ulozi do promenne Count2
*/
  for (Count2 = 0; Count2 < 3; Count2++)
  {
    if (form.rad2[Count2].checked)
      break;
  }
/*
zvysime volbu zaskrtnuteho tlacitka o jedna
pro praci s poly v Matlabu, v Matlabu se indexuje od 1
*/
  Count1++;
  Count2++;
//nove promenne pro uchovani vstupnich informaci
  var var1, var2, var3, var4;
  var1=Count1; // zaskrtnuta metoda
  var4=Count2; // barva okna
  var2= form.text1.value; // delka okna
  var3= form.text2.value; // delka prekryvu
//vysledek se presmeruje na stranku analyzaAVypis.mlsp
//s CGI parametry s ulozenymi vstupy
  self.open("analyzaAVypis.mlsp?var1="+ var1 +
   "&var2=" + var2 + "&var3=" + var3 +
   "&var4=" + var4+"&submit=Submit");
}
</SCRIPT>
</BODY>
<!-- formular pro vstupni hodnoty -->
<FORM NAME="jedna">
Typ analyzy : <BR><BR>
Zakladni ton reci
<INPUT TYPE="radio" NAME="rad"
Value="rad_button1" checked="true">
<BR>
Energie
<INPUT TYPE="radio" NAME="rad" Value="rad_button2">
<BR>
Pocet pruchodu nulou
```

```
<INPUT TYPE="radio" NAME="rad" Value="rad_button3">
<BR> <BR> <BR>
Barva grafu : <BR> <BR>
cerna
<INPUT TYPE="radio" NAME="rad2"
 Value="rad_button1" checked="true">
<BR>
modra
<INPUT TYPE="radio" NAME="rad2" Value="rad_button2">
<BR>
cervena
<INPUT TYPE="radio" NAME="rad2" Value="rad_button3">
<BR> <BR>
Delka okna
<INPUT TYPE="text" NAME="text1" Value="512" size="3">
<BR> <BR>
Delka prekryvu
<INPUT TYPE="text" NAME="text2" Value="256" size="3">
<BR> <BR>
<INPUT TYPE="button" NAME="button" Value="Click"
                onClick="vyhodnot(this.form)">
<BR> <BR>
</FORM>
</HTML>
```

B.7 Program - Výpis výsledků

Obsah souboru textttanalyzaAVypis.mlsp :

```
<html>
<head>
<%@ include file="/Scripts/header.inc" %>
<title>Vysledek analyzy</title>
</head>
<body>
<matlab:Engine>
  <!-- ulozi vstupni parametry do promennych v Matlabu -->
<matlab:Command cmd="a=${param.var1}"/><!-- zvolena analyza -->
<matlab:Command cmd="b=${param.var2}"/><!-- delka okna -->
```

```
<matlab:Command cmd="c=${param.var3}"/><!-- delka prekryvu -->
<matlab:Command cmd="d=${param.var4}"/><!-- barva -->
  <!-- nacteni nahraneho souboru do Matlabu pro analyzu -->
<matlab:Command  cmd="[vstup Fvz] = wavread('soubor.wav')"/>
<!-- spusteni funkce pro analyzu -->
    <matlab:Command cmd="
        [F0 F0param E Eparam ZCR ZCRparam vysledek]=
                    vypocet_priznaku(vstup,Fvz,b,c);"/>
  <!-- ulozeni vysledku do jedne matice -->
<matlab:Command cmd="vsechnyHodnoty=[F0 ;E; ZCR]"/>
  <!-- ulozeni potrebnych vysledku dle zvolene analyzy do matice
   hodnotyGraf, pro dalsi vykresleni charakteristiky -->
<matlab:Command cmd="hodnotyGraf=vsechnyHodnoty(a,:)"/>
  <!-- podle volby ulozime do promenne barva zvolenou barvu -->
<matlab:Command cmd="if(d==1)barva='k'; end;
                        if(d==2)barva='b'; end;
                        if(d==3)barva='r'; end;"/>
  <!-- popisek osy x -->
<matlab:Command cmd="osaXText='n  [-]'"/>
  <!-- nastaveni legendy a popisku osy y dle zvolene analyzy -->
<matlab:Command cmd="
if(a==1)legendaText='Zakladni ton reci';osaYText='F0  [Hz]';end;
if(a==2)legendaText='Energie'; osaYText='E  [-]'; end;
if(a==3)legendaText='Pruchody nulou';osaYText='E  [-]';end;
"/>
<!-- vykresleni charakteristiky
 soubor pojmenujeme jako kombinaci cisel metody,
 delky okna,delky prekryvu a barvy -->
<matlab:PlotData cmd=" plot(hodnotyGraf,'Color', barva);
       xlabel(osaXText); ylabel(osaYText);
       legend(legendaText); grid on;" handle="h1"
       filename="${param.var1}+${param.var2}+
       ${param.var3}+${param.var4}">
<matlab:Thumbnail
imagename="${param.var1}+${param.var2}+${param.var3}+
            ${param.var4}" width="1024" height="768"
        />
</matlab:PlotData>
  <!-- vypis pozadovanych vysledku -->
```

```
<matlab:Command cmd="hodnoty=vysledek(:,a)"/>
<matlab:Command cmd="h1=hodnoty(1)"/>
<matlab:Command cmd="h2=hodnoty(2)"/>
<matlab:Command cmd="h3=hodnoty(3)"/>
<matlab:Command cmd="h4=hodnoty(4)"/>
<br>Stredni hodnota     : <matlab:WriteData name="h1"/>
<br>Maximalni hodnota   : <matlab:WriteData name="h2"/>
<br>Minimalni hodnota   : <matlab:WriteData name="h3"/>
<br>Smerodatna odchylka : <matlab:WriteData name="h4"/>
</matlab:Engine>
</body>
</html>
```

B.8 Program - MFile s analýzou

Obsah souboru textttvypocet priznaku.m :

```
function ...
[F0 F0param E Eparam ZCR ZCRparam vysledek] = ...
vypocet_priznaku(vstup,Fvz,winlen,overlen)
F0=getpitch(vstup,Fvz,winlen,overlen);
E=getenergy(vstup,Fvz,winlen,overlen);
ZCR=getzcr(vstup,winlen,overlen);
F0param(1)=mean(F0);
F0param(2)=max(F0);
F0param(3)=min(F0);
F0param(4)=std(F0);
Eparam(1)=mean(E);
Eparam(2)=max(E);
Eparam(3)=min(E);
Eparam(4)=std(E);
ZCRparam(1)=mean(ZCR);
ZCRparam(2)=max(ZCR);
ZCRparam(3)=min(ZCR);
ZCRparam(4)=std(ZCR);
vysledek= [F0param(:) Eparam(:) ZCRparam(:) ]

%% funkce pro segmentaci reci
function f=enframe(x,win,inc)
```

```
nx=length(x(:));
nwin=length(win);
if (nwin == 1)
   len = win;
else
   len = nwin;
end
if (nargin < 3)
   inc = len;
end
nf = fix((nx-len+inc)/inc);
f=zeros(nf,len);
indf= inc*(0:(nf-1)).';
inds = (1:len);
f(:) = x(indf(:,ones(1,len))+inds(ones(nf,1),:));
if (nwin > 1)
    w = win(:)';
    f = f .* w(ones(nf,1),:);
end

%% funkce pro vypocet zakladniho tonu
function pitch=getpitch(input,Fvz,winlen,overlen)
Y=enframe(input,winlen,overlen);
Y=Y';
[M N]=size(Y);
pitch(1)=100;
for i=2:N
   pitch(i)=get_pitch2(Y(:,i),Fvz,pitch(i-1));
end
for i=1:length(pitch)
if pitch(i)<85
       pitch(i)=0;
end
end

function pitch = get_pitch2(s,fs,previous_pitch)
s(length(s)+1:8192) = zeros(1,8192-length(s));
S = fft(s);
f_step = fs/length(s);
```

```
NMAX = ceil(450/f_step);
NMIN = floor(50/f_step);
NMID = floor(previous_pitch/f_step);
window = zeros(4*NMAX,1);
WMID = floor(length(window)/2);
wl = ceil(2.5*NMID);
window(WMID-floor(wl/2):WMID-floor(wl/2)+wl-1)= hamming(wl);
F = S(1:NMAX);
F = F.*S(2:2:2*NMAX);
F = F.*S(3:3:3*NMAX);
F = F.*S(4:4:4*NMAX);
F = F.*window(WMID-NMID:NMAX+WMID-NMID-1);
[m i] = max(F(NMIN:NMAX));
i = i + NMIN - 1;
pitch = f_step*(i-1);

%% funkce pro vypocet energie
function E=getenergy(vstup,Fvz,winlen,overlen)
Y=enframe(vstup,winlen,overlen);
Y=Y';
E=sum((Y).^2);

%% funkce pro vypocet pruchodu nulou
function zc=getzcr(vstup,winlen,overlen)
Y=enframe(vstup,winlen,overlen);
Y=Y';
YY=sign(Y);
[M N]=size(Y);
zcc=0;
for i=1:N
    for j=2:M
zcc=zcc+abs(YY(j,i)-YY(j-1,i));
    end
  zc(i)=zcc;
  zcc=0;
end
```

C TŘETÍ PŘÍLOHA

C.1 Obsah CD

1. Apache Tomcat – instalační balíček Apache Tomcat verze 6.0.18
2. Elektonická verze dokumentu – PDF dokument s elektronickou verzí práce
3. Java Decompiler – dekompilátor Java kódu (verze jádra 0.4.7, verze grafického rozhraní 0.2.8) = možno dekompilovat všechny soubory typu **.class**
4. JDK – instalační balíčky JSDK verze 1.6 a 1.7
5. JMatLink – knihovna JMatlink.dll pro spojení Javy a Matlabu
6. MSP – instalační balíček MatlabServer Pages od A. Kizila
7. NetBeans 6.5 – instalační balíček Netbeans IDE verze 6.5
8. PostCast Server – instalační balíček free edice PostCast Serveru verze 2.6.0
9. Webový archív s aplikací – webový archív s aplikací č.2