

Web-aplikace pro identifikaci systémů z experimentálních dat

Bc. Petr Mičola

Diplomová práce
2014



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2013/2014

ZADÁNÍ DIPLOMOVÉ PRÁCE (PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Petr Mičola**
Osobní číslo: **A12859**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Automatické řízení a informatika**
Forma studia: **kombinovaná**

Téma práce: **Web-aplikace pro identifikaci systémů
z experimentálních dat**

Zásady pro vypracování:

1. Vypracujte literární rešerši na dané téma.
2. Navrhněte a realizujte webovou aplikaci, která na základě nahraných vstupně-výstupních dat z procesu umožní tento identifikovat lineárním diskrétním modelem s uživatelem definovanou strukturou.
3. Aplikaci koncipujte jako dvojjazyčnou (CZ/ENG) a pro její vytvoření využijte platformu Microsoft .NET spolu s programovým systémem MATLAB.
4. Zabezpečte, aby se v aplikaci dala využít experimentální data ze souborů typu MAT, TXT a XLS.
5. Umožněte jednoduchou filtraci dat, přepočet výsledného modelu na jeho spojité vyjádření, export výsledků do programového systému MATLAB a zobrazení vybraných statistických charakteristik.
6. Vytvořenou aplikaci důkladně otestujte a porovnejte získané výsledky s jiným přístupem.
7. Věnujte dostatečnou pozornost zabezpečení aplikace.

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. BOBÁL, Vladimír. Identifikace systémů. Zlín: Univerzita Tomáše Bati ve Zlíně, 2009. ISBN 978-80-7318-888-7.
2. RAKUS, David. Web-aplikace pro přímý návrh a ladění regulátorů z experimentálních dat. Zlín, 2009. Diplomová práce. Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky.
3. GAZDOŠ, František a David RAKUS. MAT Server: Přímý návrh a ladění regulátorů z experimentálních dat [online]. c2009-2010 [cit. 2014-01-08]. Dostupné z: <http://matserver.utb.cz>.
4. PÍSEK, Slavoj. ASP.NET (začínáme programovat). Praha: Grada Publishing, 2003. ISBN 80-247-0526-5.
5. MICROSOFT. Vytváříme zabezpečené aplikace v Microsoft ASP.NET. Praha: Computer Press, 2004. ISBN 80-251-0466-4.
6. KARBAN, Pavel. Výpočty a simulace v programech Matlab a Simulink. Praha: BEN-technická literatura, 2007. ISBN 978-80-251-1448-3.
7. ZEMEK, Lukáš. Bezpečnost webových aplikací. Praha, 2012. 68 s. Bakalářská práce. Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky.

Vedoucí diplomové práce:

doc. Ing. František Gazdoš, Ph.D.

Ústav řízení procesů

Datum zadání diplomové práce:

7. března 2014

Termín odevzdání diplomové práce:

11. června 2014

Ve Zlíně dne 7. března 2014

prof. Ing. Vladimír Vašek, CSc.
děkan



prof. Ing. Vladimír Vašek, CSc.
Každitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- Že odevzdaná verze diplomové/bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

ABSTRAKT

V této diplomové práci je nejprve stručně popsána problematika internetových aplikací, kde jsou popsány základní vlastnosti technologií použitých v této diplomové práci. Dále je obecně popsán programový systém MATLAB a je zde popsán způsob jakým lze tento systém napojit na internetové aplikace. Stručně popsána je také metoda nejmenších čtverců, která je využita při identifikaci modelu z naměřených experimentálních dat. Hlavním cílem práce bylo vytvoření internetové aplikace, která umožňuje identifikaci z experimentálních dat. Zmíněného cíle bylo dosaženo vytvořením aplikace pomocí *frameworku* ASP.NET v kombinaci s programovacím jazykem C#. Identifikační algoritmy byly naprogramovány v prostředí MATLAB a následně nainportovány pomocí MATLAB Compileru do internetové aplikace.

Klíčová slova: Internetové technologie, Internetová aplikace, identifikace, C#, Microsoft Visual Studio, MATLAB Builder NE, ASP.NET

ABSTRACT

Firstly the issues of internet applications are described. Then basic features of technologies used in this thesis are discussed. Furthermore the MATLAB software system is briefly introduced. The way how to connect the MATLAB to Internet applications is shown and the least squares method is described briefly as well. The method is used in the identification of the model from the measured experimental data. The main goal was to create a web application that allows the identification of the experimental data. This was achieved by creating a web application using ASP.NET framework in combination with the programming language C#. Identification algorithms were programmed in MATLAB and built using the MATLAB Compiler. Resultant libraries outputs of the compiler were finally included into the internet application.

Keywords: Internet technologies, Internet application, identification, C#, Microsoft Visual Studio, MATLAB Builder NE, ASP.NET

Poděkování

Tímto bych rád poděkoval panu doc. Ing. Františkovi Gazdošovi, Ph.D., který jeho věcnými připomínkami a odbornými radami přispěl k vypracování této diplomové práce.

Motto

„Šťěstí přeje silným a statečným.“

„Talent není všechno. Ta dřina za to stojí.“

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČÁST.....	10
1 INTERNETOVÉ APLIKACE	11
1.1 WEBOVÝ SERVER	11
1.2 VÝVOJOVÉ PROSTŘEDÍ INTERNETOVÝCH APLIKACÍ	11
1.3 STRUKTURA INTERNETOVÝCH APLIKACÍ	12
2 TECHNOLOGIE V INTERNETOVÉ APLIKACI.....	13
2.1 PREZENTAČNÍ VRSTVA	13
2.1.1 HTML	13
2.1.2 CSS.....	13
2.1.3 AJAX.....	14
2.2 LOGICKÁ VRSTVA.....	14
2.2.1 ASP.NET.....	14
2.3 DATOVÁ VRSTVA	15
2.3.1 Souborový systém	16
2.3.2 Databáze.....	16
3 PROGRAMOVÉ PROSTŘEDÍ MATLAB	17
3.1 ROZŠÍŘENÍ SYSTÉMU MATLAB.....	17
3.2 NEKOMERČNÍ ALTERNATIVY	18
4 VYUŽITÍ MATLAB KNIHOVEN V INTERNETOVÝCH APLIKACÍCH	19
4.1 MATLAB COMPILER	19
4.2 PROPOJENÍ KOMPONENT S INTERNETOVOU APLIKACÍ	21
5 IDENTIFIKACE SYSTÉMŮ	23
5.1 VOLBA MODELU	23
5.2 METODA NEJMENŠÍCH ČTVERCŮ.....	25
6 AUTOMATICKÉ PLÁNOVÁNÍ ÚLOH VE WINDOWS.....	29
II PRAKTICKÁ ČÁST	30
7 NASTAVENÍ WEBOVÉHO SERVERU	31
7.1 KONFIGURACE WEBOVÉHO SERVERU	31
7.2 VÝVOJOVÉ PROSTŘEDÍ PRO TVORBU INTERNETOVÝCH APLIKACÍ	32
7.3 MATLAB NA SERVERU	32
8 UŽIVATELSKÉ ROZHRAŇÍ INTERNETOVÉ APLIKACE	33

8.1	ÚVOD	33
8.2	VÝBĚR A IMPORT SOUBORU.....	33
8.3	URČENÍ PROMĚNNÝCH Z IMPORTOVANÉHO SOUBORU	34
8.4	FILTRACE IMPORTOVANÝCH DAT	35
8.5	ZVOLENÍ STUPŇŮ POLYNOMŮ ARX MODELU A VÝPOČET.....	36
8.6	SROVNÁNÍ IDENTIFIKOVANÉHO MODELU S UŽIVATELSKÝMI DATY.....	37
8.7	ZMĚNA JAZYKA	39
8.8	UŽIVATELSKÉ TESTOVÁNÍ	40
9	KÓDOVÉ POZADÍ VYTVOŘENÉ APLIKACE.....	41
9.1	STRUKTURA INTERNETOVÉ APLIKACE	41
9.2	M-SOUBORY POUŽITÉ PRO VYTVOŘENÍ KOMPONENTY.....	43
9.2.1	Soubory mFilterN.m, mFilterP.m	43
9.2.2	Soubory mLoad.m, mLoadTXT.m, mLoadXLS.m.....	43
9.2.3	Soubory mPlot.m, mPlotMulti.m	44
9.2.4	Soubory sD2C.m, sDen.m, sNum.m	44
9.2.5	Soubor sSim.m	44
9.2.6	Soubor sModel.m	45
9.2.7	Export naprogramovaných m-souborů do dll knihoven.....	45
9.3	SOUBORY CS, ASPX, ASCX	46
9.3.1	Soubor Default.aspx	46
9.3.2	Soubor mod_ident.ascx	47
9.4	LOKALIZACE APLIKACE.....	49
9.5	CSS INTERNETOVÉ APLIKACE.....	50
9.6	OVĚŘENÍ UŽIVATELSKÝCH VSTUPŮ	51
10	SROVNÁNÍ VÝSLEDKŮ APLIKACE S VÝSLEDKY Z MATLABU.....	52
10.1	GENEROVÁNÍ TESTOVACÍCH DAT	52
10.2	POROVNÁNÍ VÝSLEDNÝCH DISKRÉTNÍCH PŘENOSŮ	52
10.3	GRAFICKÉ POROVNÁNÍ PRŮBĚHŮ	53
11	AUTOMATICKÉ MAZÁNÍ STARÝCH UŽIVATELSKÝCH DAT	54
11.1	SKRIPT PRO MAZÁNÍ STARÝCH SOUBORŮ.....	54
11.2	ZAPLÁNOVÁNÍ SKRIPTU	55
	ZÁVĚR	59
	SEZNAM POUŽITÉ LITERATURY.....	61
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	63
	SEZNAM OBRÁZKŮ	64
	SEZNAM PŘÍLOH.....	66

ÚVOD

V dnešní době se stále více služeb a prezentací přesouvají na internet. Důvodů k tomuto kroku je celá řada. Zřejmě největší výhodou je dostupnost odkudkoli na světě, kde je dostupné datové připojení. Nespornou výhodou je nezávislost na uživatelském vybavení, jelikož jedinou důležitou podmínkou je internetový prohlížeč.

Za posledních několik let klesá cena hardwarových prostředků a internetového připojení. Opačným směrem naopak roste datová rychlost a datové limity, které se stále zvedají. Pravděpodobně tedy není lepšího prostředku, jak uživatelům po celém světě prezentovat stejný obsah v jediný okamžik.

Tato diplomová práce se zaměřuje na vývoj webové aplikace pro identifikaci z experimentálních dat využívající ASP.NET *frameworku*, programovacího jazyka C# a programového systému MATLAB. Podobně zaměřená webová aplikace již vznikla v rámci diplomové práce pana Ing. Rakuse dostupná na <http://matserver.utb.cz/>. Zmíněná diplomová práce slouží k přímému návrhu a ladění regulátorů z experimentálních dat. Tato aplikace na základě nahraných vstupně-výstupních dat navrhne lineární jednorozměrný regulátor s uživatelem definovanou strukturou. Řešená i zmíněná aplikace využívají stejných programových prostředků a po menších úpravách ze strany vývojáře by byly obě aplikace kompatibilní. V aplikaci řešené touto DP je hlavním cílem identifikace modelu z uživatelem nahraných experimentálních dat a ve zmíněné aplikaci p. Rakuse je to návrh a ladění regulátoru. Další rozdíly jsou především v podpoře typů importovaných souborů, filtraci dat a multijazyčnosti aplikace. V této diplomové práci tedy bylo využito některých poznatků a postřehů z aplikace pana Ing. Rakuse [2].

Řešená webová aplikace využívá stávající webový server, na kterém byla přidána další aplikace. Webová aplikace byla naprogramována zmíněnými internetovými technologiemi a je dostupná na subdoméně <http://matserver.utb.cz/ExpIdent/>.

I. TEORETICKÁ ČÁST

1 INTERNETOVÉ APLIKACE

Internetovou neboli webovou aplikaci lze chápat jako službu, která je přístupná odkudkoli na světě za podmínky internetového připojení klienta. Za této podmínky se jedná o velmi silný nástroj, jehož možnosti jsou široké. Již dnes je jejich rozsah opravdu velký a dle mého názoru jejich význam poroste stále větším tempem. Velkou roli v tomto ohledu hraje i stále se zvyšující dostupnost a rychlost internetového připojení. V kombinaci s klesající cenou budou tyto aplikace nevyhnutelnou součástí našeho moderního života.

1.1 Webový server

Internetová aplikace je umístěna na vyhrazeném počítači, kde běží. Takovýto počítač se nazývá webový server. Nejčastější technologií využívanou pro vytvoření webového serveru je operační systém Linux s webovým serverem apache nebo nginx. Další velmi rozšířenou variantou je Microsoft Windows server s jeho IIS serverem.

Klienti pak přistupují prostřednictvím internetového prohlížeče k dané aplikaci. V praxi to pak vypadá tak, že klient posílá HTTP požadavky na webový server, který tyto požadavky zpracuje a vrátí odpověď klientovi. Klient odpověď zpracuje prostřednictvím internetového prohlížeče a výsledný obsah pak zobrazí uživateli.

Internetové prohlížeče jako interpreti webového obsahu používají jiná vykreslovací jádra. Z tohoto důvodu se může občas stát, že obsah internetové aplikace nebude vypadat ve všech prohlížečích stejně. Je potřeba na tuto skutečnost myslet a uživatele upozornit na to pro jaký prohlížeč byla aplikace optimalizována.

1.2 Vývojové prostředí internetových aplikací

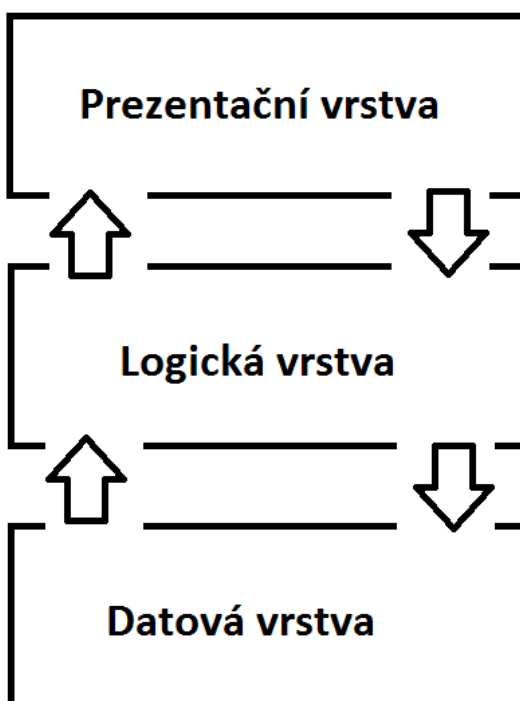
Internetové aplikace se dají programovat v podstatě v jakémkoli textovém editoru např. poznámkový blok, Microsoft Word,... Avšak takové editory nejsou vůbec vhodné a komfortní směrem k vývoji internetových aplikací. Není k tomu ani důvod, jelikož k takovému účelu nebyly vytvořeny. Pro tyto ale i mnohé další důvody vznikly vývojové prostředí, které usnadňují vytváření internetových aplikací. Příkladem může být *Visual studio*, *PS Pad*, *Sublime Text*, a mnoho dalších. Každý vývojář si pak vybere takové prostředí, které mu pro daný typ aplikace sedí a vyhovuje nejlépe.

Dříve se psaly internetové aplikace přímo v daném programovacím jazyce a využívalo se jen systémových knihoven. Dnes je již toto minulostí a nevyplatí se z ekonomického

hlediska psát si vlastní knihovny, které již jsou někým naprogramovány. Proto vznikla celá řada tzv. *frameworků*. Díky těmto „knihovnám“ se programátor dopouští méně chyb a využívá již dříve naprogramované funkce v daném *frameworku*. Po určitém seznamovacím čase je i výkonost programátora značně vyšší.

1.3 Struktura internetových aplikací

Většina internetových aplikací se pomyslně rozděluje do tří vrstev. Jednotlivé vrstvy spolu souvisí a komunikují spolu. Jedná se o vrstvy prezentační, logickou a datovou. Prezentační vrstva je tvořena internetovým prohlížečem. Logická vrstva slouží pro dynamické generování stránek např. PHP, ASP,... Datová vrstva je obvykle tvořena databázemi. Komunikace pak vypadá tak, že internetový prohlížeč posílá žádosti logické vrstvě, která posílá požadované dotazy do databáze. [2]



Obrázek 1: struktura internetové aplikace

2 TECHNOLOGIE V INTERNETOVÉ APLIKACI

V této části kapitoly budou stručně popsány použité technologie v jednotlivých vrstvách internetové aplikace.

2.1 Prezentační vrstva

Prezentační vrstvou tedy rozumíme jakýsi prostředek, díky kterému zobrazujeme data uživateli. Jedná se především grafické prvky, uživatelské rozhraní. U internetových aplikací je k tomu ve většině případů použit internetový prohlížeč. V této se jen zobrazují dané prvky. Nejčastěji je internetový obsah tvořen pomocí HTML jazyka doplněným o patřičné nastýlování pomocí CSS, případně javascriptu, flash,... [2]

2.1.1 HTML

HTML je hypertextový značkovací jazyk, který se používá pro tvorbu internetových stránek. Díky jeho vzniku již nebyly dokumenty na internetu již pouhé textové soubory, ale měly již pro uživatele přívětivější tvar. Dříve se hojně používala verze HTML 4.01. Z čistého HTML se postupně vyvinul XHTML a HTML5.

HTML používá tzv. tagy a atributy, které jsou podporovány nejrozšířenějšími prohlížeči Internet Explorer, Mozilla, Opera, Google Chrome atd... Pomocí těchto tagů a atributů je rozdělen internetový dokument do několika částí, kterým pak internetový prohlížeč rozumí a umí je uživateli zobrazit.

[10]

2.1.2 CSS

Vzhledem k nedostatečné podpoře a složitější úpravě formátování internetového obsahu přímo v HTML vznikly CSS. Díky této technologii se stala úprava formátování dokumentu čistší a přehlednější. Neboť jinak byly do kódu stránky zapisovány i atributy ohledně formátování a to je značně nevhodné. CSS jsou většinou uloženy mimo hlavní dokument a jsou do HTML dokumentu připojeny externě.

CSS nabízí celou řadu možností jak formátovat internetový obsah. Nemá smysl vyjmenovávat všechno co lze pomocí CSS udělat, ale například uvedu velikost textu, pozicování, úprava pozadí, obtékání, úprava barev a mnoho dalšího.

[10]

2.1.3 AJAX

Ajax je technologie díky, které není nutné znovu načítat kompletně celý obsah internetové stránky. V mnoha případech stačí načíst pouze změněnou část a zbytek obsahu zůstane stejný. Jedná se o takzvané asynchronní zpracování. Nejčastěji je implementováno řešení pomocí knihoven napsaných v Javascriptu.

Funkce napsané v Javascriptu tedy běží na straně klienta, tudíž v internetovém prohlížeči uživatele. Jednotlivé skripty se pak zpracovávají přímo na straně klienta a ne serveru a tímto se pak také šetří samotný výkon serveru. Na druhou stranu tím může trpět výkon uživatelského internetového prohlížeče.

Největší nevýhodou je možnost zcela vypnout podporu Javascriptu v prohlížeči. Díky různým verzím prohlížečů a jejich vykreslovacím jádrům můžeme docházet k odlišnostem a stejná funkce nemusí vždy fungovat stejně.

Na druhou stranu jeho velkou výhodou je rozšířenost a dynamičnost. Pomocí Javascriptu lze oživit internetové stránky a dodat jim příjemnější vzhled, obohatit je o interaktivní prvky, atd...

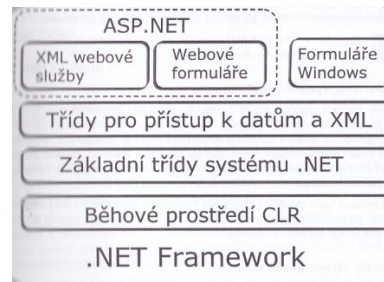
[10]

2.2 Logická vrstva

V třívrstvě modelu se používá logická vrstva zejména pro komunikaci mezi prezentační a datovou vrstvou. Obsahuje zejména algoritmy, které určují jádro aplikace. Tato vrstva řídí celou aplikaci. Běžný uživatel by neměl k ní mít přístup a neměl by poznat, že existuje. Zde se zpracovávají požadavky uživatele nebo jiných služeb. k programování se využívají zejména programovací jazyky a technologie jako PHP, ASP.NET, Perl,... [2]

2.2.1 ASP.NET

ASP.NET je technologie, která je součástí systému .NET frameworku. Díky tomuto systému může programátor využívat soubor objektů a funkcí, které jsou nezbytné pro bezpečný vývoj a běh programů. Systém se skládá z následujících částí zobrazených na obrázku níže.



Obrázek 2: .NET framework

Běžové prostředí CLR slouží ke spuštění programů a přiděluje/uvolňuje jim prostředky. Při zkompileování napsaného programu dojde k přeložení na tzv. metadata. Tyto pak slouží k reprezentaci daného kódu a nesou informaci co má aplikace dělat. Program se pak spustí přes běžové prostředí CLR, které přeloží metadata do strojového kódu.

Základní třídy systému .NET byly vytvořené k zjednodušení práce programátorů. Tyto třídy pak obsahují objekty, funkce, metody například pro práci s číselnými datovými typy *Integer*. Formuláře uživatelského rozhraní slouží k tvorbě UI vytvořených aplikací. První částí jsou aplikace určené pro systém Windows, kde pak běží na daném počítači. Druhou částí jsou objekty pro vytvoření internetových aplikací, které běží přímo na webovém serveru. Této druhé části se pak říká ASP.NET. Pak je tedy zřejmé, že ASP.NET není programovací jazyk, ale soubor funkcí a objektů, které dovolují programovat webové stránky běžící na serveru. Internetové stránky se pak můžou programovat pomocí jakéhokoli jazyka např. C#, *Visual Basic* .NET, C++,...

ASP.NET je založen na modelu klient/server. Stručně to lze tedy popsat tak, že uživatel zadá požadavek na server a ten vrátí HTML kód prohlížeči klienta. ASP.NET podporuje i událostmi řízený model, kdy je např. monitorován stisk tlačítka a podle toho vykreslit danou stránku. Pro tento typ událostí lze použít i Javascript, který běží na straně klienta. Mezi hlavní výhody ASP.NET patří skutečnost, že je kompilovaný pomocí CLR do strojového kódu počítače a tím zaručuje vyšší rychlost provádění kódu. Dále je třeba zmínit interaktivnost stránek, možnost využití různých programovacích jazyků, snadné vytváření UI.

[3] [4] [9] [15]

2.3 Datová vrstva

Tato vrstva slouží nejčastěji k uchování uživatelských a jiných dat. Zde se ukládají a čtou potřebná data. Velkou výhodou je oddělení datové vrstvy od prezentační

vrstvy. Neboť změna použité technologie v logické vrstvě je pak méně náročná na přepis aplikace. Data se mohou ukládat buď do souborového systému, nebo přímo do databáze.

2.3.1 Souborový systém

Označení způsobu organizace dat ve formě souborů a adresářů, takovým způsobem, aby k nim byl umožněn jednoduchý přístup. Souborové systémy mohou být uloženy na různém druhu elektronické paměti a to jak přímo na pevném disku nebo přenositelném médiu, tak na vzdáleném síťovém serveru. Souborových systémů je celá řada a každý operační systém upřednostňuje jiný. Každý souborový systém je vhodný na dané konkrétní využití.

[11]

2.3.2 Databáze

Jedná se o uspořádanou množinu informací, která je uložena na paměťovém médiu. Někdy je pod pojmem databáze chápán i systémový prostředek pomocí, kterého se přistupuje a manipuluje s daty. Podobně jako u souborového systému, tak i databází je mnoho druhů. k různým účelům se používají jiné druhy databází.

[12]

3 PROGRAMOVÉ PROSTŘEDÍ MATLAB

Dnes lze MATLAB považovat za jeden z nejrozšířenějších programovacích balíčků pro vědecko-technické výpočty v mnoha oborech. Tento komplexní systém je využíván pro výuku na mnoha technických fakultách po celé republice. Jeho aplikace se však používá i ve firmách a výzkumných pracovištích.

MATLAB je tedy velmi výkonné interaktivní prostředí, které slouží pro celou řadu výpočtů ať už z oblasti techniky, vědy nebo finanční analýzy. Lze ho také využít k vizualizaci dat nebo přímo jako programovací jazyk.

Jedna z velkých výhod, kterou MATLAB disponuje, je možnost doinstalování a využití celé řady modulů. Díky nimž najde uplatnění nejen u jedné cílové skupiny, ale můžou ho využít jak technici, vědci, matematici tak učitelé při řešení nejrůznějších problémů. Další nesmírnou výhodou, která je opravdu velmi důležitá a v mnoha případech hraje hlavní roli, je výborná dokumentace použitých funkcí i s následnou vizualizací. U takového systému samozřejmě nesmí chybět ani oficiální fórum uživatelů.

MATLAB disponuje opravdu obrovským množstvím vestavěných funkcí, které značně zjednodušují řešení problémů, vývoj aplikací, testování nejrůznějších modelových situací,...

Hlavními výhodami tohoto komplexního systému bezesporu jsou fakta, že se jedná o vysokoúrovňový a rozšiřitelný systém, velké množství aplikačních knihoven, podpora vícerozměrných polí a datových struktur, interaktivní nástroje pro tvorbu GUI, import a export do mnoha formátů, možnost komunikace s externími měřicími a monitorovacími přístroji v reálném čase, rozšiřitelnost modulů jazyky C, C++, Fortran, Java,... [5]

3.1 Rozšíření systému MATLAB

Jednou z velkých výhod systému MATLAB je možnost rozšíření o doplňující moduly tzv. toolboxy, díky kterým se práce a řešení problémů stane ještě jednodušším a intuitivnějším.

MATLAB lze standardně rozšířit například o následující komponenty pro matematické výpočty a optimalizace, statistika analýza dat, návrh a analýza řídicích systémů, simulace systémů a procesů, zpracování signálů a komunikace, zpracování obrazu, měření a testování, finanční analýza a modelování, tvorba aplikací.

Simulink je nejrozšířenější rozšíření pro MATLAB. Využíváno je zejména k simulaci dynamických systémů. Tento doplněk využívá algoritmy z MATLABu pro řešení nelineárních diferenciálních a diferenčních rovnic. V tomto programu se pracuje s blokovými schémata a jejich následnému propojení. [5]

3.2 Nekomerční alternativy

Alternativou ke komerčnímu systému MATLAB může být GNU Octave. Jedná se o podobný vysokoúrovňový jazyk, který byl vytvořen pod licencí GPL. Pod touto licencí jsou zdarma přístupné zdrojové kódy i binární distribuce jak pro operační systém Linux tak pro Windows.

Od počátku vývoje tohoto projektu již byla implementována celá řada toolboxů z Matlabu např. optimalizační toolbox, toolbox pro zpracování signálu, ...

Dalším nekomerčním řešením může být Scilab, který je také dostupný na většinu běžných operačních systémů.

[5]

4 VYUŽITÍ MATLAB KNIHOVEN V INTERNETOVÝCH APLIKACÍCH

V dřívějších verzích Matlabu (do R2006b) byl obsažen MATLAB Web Server. Díky němuž byla možná komunikace mezi webovým serverem a Matlabem. Zdrojové soubory se nahrávaly přímo na web server. Tímto způsobem se velmi jednoduše začlenil Matlab do internetových HTML stránek a byl dostupný širokému spektru uživatelů.

Rozšíření MATLAB Web server však již dále není podporována a tudíž ani není vhodné ji dále používat. Jeho vývoj byl ukončen společností vyvíjející MATLAB.

Místo Web serveru byly vytvořeny nové možnosti, jak využít MATLAB ve webových aplikacích. Nyní se využívá tzv. komponent neboli dll knihoven. To je užitečné zejména při paralelní práci více vývojářů, kdy každý dělá na své části a na konci projektu se jednoduše sloučí části do jednoho celku. Vývojáři vytváří své vlastní M-soubory, které obsahují standardní příkazy a funkce systému MATLAB. Tyto soubory se pak převedou na zmíněné komponenty, které se pak naimportují do separátně vyvíjené webové aplikace.

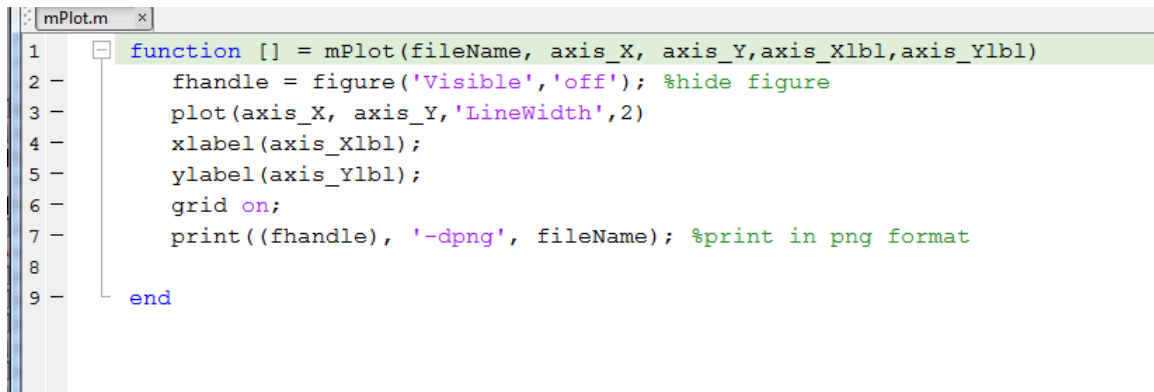
Benefitem tohoto přístupu vytváření webových aplikací je možnost použití různých technologií pro programování internetové aplikace jako například .NET Framework, JAVA,...

[2]

4.1 MATLAB compiler

Toto rozšíření umožňuje vytvořit samostatné aplikace, které nebudou závislé na Matlabu. Díky tomuto toolboxu je možné vytvořit aplikaci, která může být šířena mezi zákazníky. Tyto aplikace pak využívají runtime Matlabu, který je zdarma a je volně šiřitelný.

Základem je vytvoření zdrojového M-souboru, který obsahuje vlastní kód napsaný v jazyce MATLAB. Tento zdrojový soubor je vytvořen jako funkce a pak se tímto způsobem také volá. Jednoduchá funkce na vykreslení grafu ze vstupních hodnot může vypadat následovně:

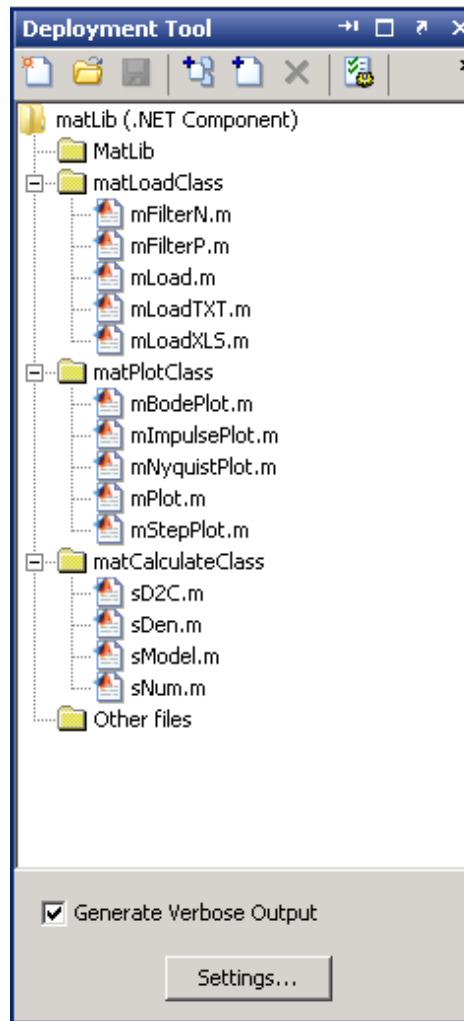


```
mPlot.m x
1 function [] = mPlot(fileName, axis_X, axis_Y,axis_Xlbl,axis_Ylbl)
2     fhandle = figure('Visible','off'); %hide figure
3     plot(axis_X, axis_Y,'LineWidth',2)
4     xlabel(axis_Xlbl);
5     ylabel(axis_Ylbl);
6     grid on;
7     print((fhandle), '-dpng', fileName); %print in png format
8
9     end
```

Obrázek 3: Zdrojový M-soubor pro funkci na vykreslení grafu ze zadaných dat

Zdrojové soubory jsou pomocí MATLAB compileru převedeny do dll knihoven. Tento nástroj využívá pro kompilaci, v závislosti na tom pro jakou internetovou aplikaci je knihovna určena, daný *builder*. Pro .NET *framework* se využívá MATLAB *Builder* NE. Tento *Builder* vytvoří knihovny, které se mohou naimportovat přímo do aplikace využívající technologii .NET *Framework*. Pro programovací jazyk JAVA se například využívá MATLAB *Builder* JA. [2]

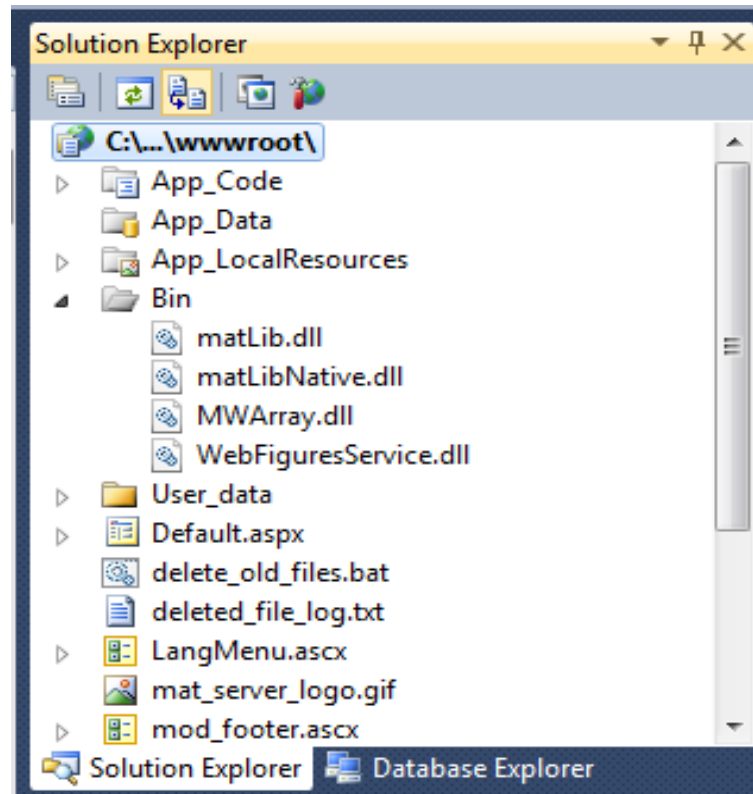
Následující obrázek ukazuje, jaká by například mohla být struktura vytvořené komponenty:



Obrázek 4: Deployment Tool

4.2 Propojení komponent s internetovou aplikací

Vytvořené dll knihovny z MATLAB *compileru* se propojují s internetovou aplikací stejným způsobem jaké každé jiné dll knihovny naimportované do .NET *Framework* aplikace. Vyvíjená aplikace neví odkud má tyto knihovny brát a proto je potřeba jí předat referenci, aby věděla odkud knihovnu načíst. Knihovnu je dále potřeba nakopírovat do složky Bin, kde jsou uloženy uživatelské knihovny. Na obrázku je znázorněná struktura aplikace i s referencí na dll knihovnu ve *Visual Studiu 2010 Express Edition*.



Obrázek 5: Složka Bin a reference na MATLAB knihovnu

MATLAB *Builder* vytvoří knihovny, které obsahují námi definované třídy. Dle použitého programovacího jazyka se pak vytváří jednotlivé instance těchto tříd a inicializují se. Pracuje se s nimi stejně jako s jinými objekty v daném programovacím jazyce. Na následujícím obrázku je vytvořena jednoduchá třída, která bude sloužit k vykreslení dle zadaných parametrů.

```
matPlotClass mPloter = new matPlotClass();  
mPloter.mPlot(sFileLocationYT, mwnaT, mwnaY,xlabel,ylabel);
```

Obrázek 6: vytvoření instance třídy z importované knihovny

5 IDENTIFIKACE SYSTÉMŮ

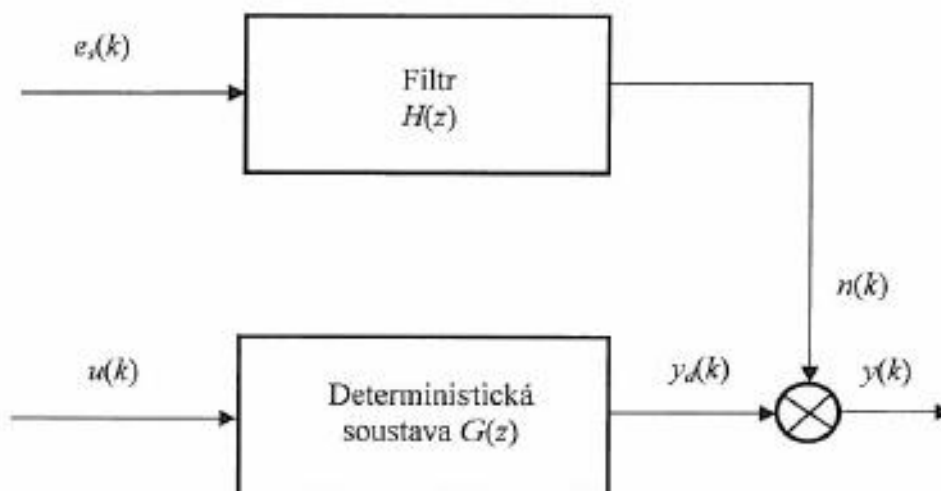
Pro efektivní řízení je třeba znát vlastnosti systému, který se chystáme řídit. Čím přesnější a detailnější informace o řízeném systému máme, tím efektivnější je pak řízení. Systémy jsou proto identifikovány pomocí matematických modelů, které odrážejí podstatné vlastnosti daného systému. Cílem je najít co nejvhodnější model odpovídající vyšetřovanému systému, tak aby chování modelu bylo srovnatelné za stejných podmínek. Avšak vždy se jedná o aproximaci, neboť okolní podmínky se stále mění. Kvalita identifikačního procesu je pak dána samotným pozorováním daného procesoru, výběrem vhodného experimentu a identifikačního algoritmu a zohledněním okolních vlivů a poruch.

[1] [7]

5.1 Volba modelu

Existuje několik druhů dynamických diskretních modelů, které jsou vhodné pro experimentální identifikaci. Rovnice obecného stochastického modelu při zanedbání dopravního zpoždění může být následující:

$$y(k) = \frac{B(z^{-1})}{A(z^{-1})F(z^{-1})} u(k) + \frac{C(z^{-1})}{A(z^{-1})D(z^{-1})} e_s(k)$$



Obrázek 7: Blokové schéma modelu obecného stochastického procesu

[1]

Deterministická soustava $G(z)$ popisuje deterministické chování stochastického dynamického objektu diskretní přenosovou funkcí:

$$G(z) = \frac{y_d(k)}{u(k)} = \frac{B(z^{-1})}{A(z^{-1})} (z^{-d}) \quad (1)$$

Výstupní veličinou z deterministické soustavy je $y_d(k)$. Účinek náhodných neměřitelných veličin je modelován pomocí náhodné veličiny $n(k)$. Z obrázku obecného stochastického procesu je zřejmé, že výstupní veličina je součtem náhodné veličiny a výstupní veličiny z deterministické soustavy. Pak tedy platí:

$$y(k) = y_d(k) + n(k) \quad (2)$$

Vyjádřením $y_d(k)$ z předešlé rovnice a dosazením do rovnice 2 bude získána rovnice:

$$A(z^{-1})y(k) = B(z^{-1})u(k)z^{-d} + A(z^{-1})n(k) \quad (3)$$

Polynomy použité v rovnici uvedené na obrázku 7 jsou ve tvaru:

$$\begin{aligned} A(z^{-1}) &= 1 + a_1 z^{-1} + \dots + a_{n_a} z^{-n_a} \\ B(z^{-1}) &= b_1 z^{-1} + b_2 z^{-2} + \dots + b_{n_b} z^{-n_b} \\ C(z^{-1}) &= 1 + c_1 z^{-1} + \dots + c_{n_c} z^{-n_c} \\ D(z^{-1}) &= 1 + d_1 z^{-1} + \dots + d_{n_d} z^{-n_d} \\ F(z^{-1}) &= 1 + f_1 z^{-1} + \dots + f_{n_f} z^{-n_f} \end{aligned}$$

Obrázek 8: Obecné tvary polynomů

Zjednodušením obecného stochastického modelu lze dostat hojně používaný model ARX ($C = D = F = 1$)

$$y(k) = \frac{B(z^{-1})}{A(z^{-1})} u(k) + \frac{1}{A(z^{-1})} e_s(k)$$

Obrázek 9: model ARX

Model ARX obsahuje společný filtr $1/A(z^{-1})$ pro deterministickou a stochastickou část modelu. Tento model patří do třídy modelů založených na chybě rovnice.

Vyjádření stochastického diskrétního modelu ve vektorovém tvaru:

$$y(k) = \Theta^T(k)\phi(k) + e_s(k) \quad (4)$$

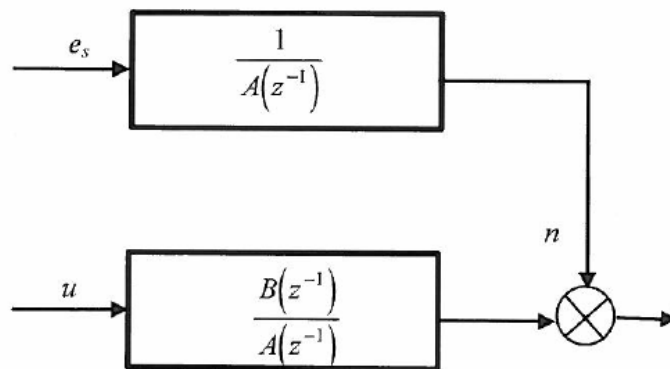
, kde $\Theta(k)$ je vektor parametrů a $\phi(k)$ je vektor dat tzv. regresor, což jsou známé funkce hodnot.

Například pro případ modelu druhého řádu s diskrétní přenosovou funkcí ve tvaru:

$$G(z) = \frac{B(z^{-1})}{A(z^{-1})} = \frac{b_1z^{-1} + b_2z^{-2}}{1 + a_1z^{-1} + a_2z^{-2}} \quad (5)$$

má pak rovnice 4 následující tvar:

$$y(k) = \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ b_1 \end{bmatrix} [-y(k-1) \quad -y(k-2) \quad u(k-1) \quad u(k-2)] + e_s(k) \quad (6)$$



Obrázek 10: Blokové schéma ARX modelu

Modely ARX jsou využívány zejména v adaptivních řídicích algoritmech, které používají průběžnou identifikaci modelu procesu v případě, že výstup identifikovaného objektu je ovlivněn náhodným poruchovým signálem typu bílého šumu. Přenosová funkce filtru je $H(z) = 1/A(z^{-1})$.

[1]

5.2 Metoda nejmenších čtverců

Jednou z nejčastěji používaných metod v identifikačních algoritmech je metoda nejmenších čtverců.

Obecně lze lineární model popsat následujícím vztahem

$$y(k) = \sum_{i=1}^r a_i f_i(k) + e(k) \quad (7)$$

, kde a_i jsou neznámé parametry f_i jsou známé funkce hodnot, chyba $e(k)$ je rozdíl naměřených hodnot $y(k)$ a modelových hodnot $y_m(k)$:

$$e(k) = y(k) - y_m(k) = y(k) - \sum_{i=1}^r a_i f_i(k) + e(k) \quad (8)$$

Odhady parametrů a_i jsou získány pomocí kritéria minimálního součtu kvadrátů chyby $e(k)$

$$J = \sum_{k=1}^N e^2(k) \quad (9)$$

Jsou-li parciální derivace kritéria, uvedeného v rovnici 9, podle jednotlivých parametrů rovny nule, pak toto kritérium nabývá minima. Odhady parametrů jsou pak dány řešením soustavy N rovnic o r neznámých.

$$y(1) = a_1 f_1(1) + \dots + a_r f_r(1) + e(1) \quad (10)$$

$$y(N) = a_1 f_1(N) + \dots + a_r f_r(N) + e(N)$$

Soustavu rovnic lze zapsat ve tvaru:

$$\mathbf{y} = \mathbf{F}\boldsymbol{\theta} + \mathbf{e} \quad (11)$$

, kde jednotlivé vektory jsou dány tvary:

$$\mathbf{y}^T = [y(1) \quad \dots \quad y(N)] \quad (12)$$

$$\boldsymbol{\theta}^T = [a_1 \quad \dots \quad a_r]$$

$$\mathbf{e}^T = [e(1) \quad \dots \quad e(N)]$$

A matice $F(N,r)$ má tvar:

$$F = \begin{bmatrix} f_1(1) & \dots & f_r(1) \\ \vdots & \ddots & \vdots \\ f_1(N) & \dots & f_r(N) \end{bmatrix} \quad (13)$$

Minimum kvadratického kritéria je získáno tak, že derivace kritéria J podle parametrů $\boldsymbol{\theta}$ je položena rovna 0

$$\mathbf{J} = \mathbf{e}^T \mathbf{e} = (\mathbf{y} - \mathbf{F}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{F}\boldsymbol{\theta}) \quad (14)$$

$$\left. \frac{\partial J}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} = 0$$

Po úpravách výsledku je získán základní maticový tvar pro jednorázový odhad metodou nejmenších čtverců:

$$\hat{\boldsymbol{\theta}} = (\mathbf{F}^T \mathbf{F})^{-1} \mathbf{F}^T \mathbf{y} \quad (15)$$

Jednou z důležitých charakteristik odhadu parametrů je jejich kovarianční matice, která popisuje přesnost výsledků. Na hlavní diagonále jsou kvadráty směrodatných odchylek a mimo hlavní diagonálu jsou kovariance (vyjadřují závislost mezi jednotlivými veličinami). Matice je dána vztahem:

$$C[\hat{\boldsymbol{\theta}}] = E[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})^T] \quad (16)$$

Za předpokladu, že chyba e je nekorelovaná platí následující vztah:

$$C[\hat{\boldsymbol{\theta}}] = \sigma_e^2 (\mathbf{F}^T \mathbf{F})^{-1} \quad (17)$$

Odhady chyby rovnice se vypočítají ze základní rovnice 11:

$$\hat{\mathbf{e}} = \mathbf{y} - \hat{\mathbf{y}} \quad (18)$$

Residuální chybová funkce je dána rovnicí:

$$J_R = \hat{\mathbf{e}}^T \hat{\mathbf{e}} \quad (19)$$

Při předpokladu že $E[e] = 0$ platí pro odhad rozptylu chyby vztah:

$$\hat{\sigma}_e^2 = \frac{J_R}{N} \quad (20)$$

Pokud budeme uvažovat obecný jednorozměrný stochastický model a předpoklady $n_a = n_b = n$; $A(z^{-1})n(k) = e_r(k)$ a zanedbání dopravního zpoždění, dostaneme z rovnice 3:

$$A(z^{-1})y(k) = B(z^{-1})u(k) + e_r(k) \quad (21)$$

Dále definujeme vektory a rovnici:

$$\mathbf{y}^T = [y(n+1), y(n+2), \dots, y(N)] \quad (22)$$

$$\mathbf{e}^T = [e_r(n+1), e_r(n+2), \dots, e_r(N)]$$

$$\hat{\boldsymbol{\theta}}^T(k) = [\hat{a}_1, \hat{a}_2, \dots, \hat{a}_n, \hat{b}_1, \hat{b}_2, \dots, \hat{b}_n]$$

, kde y^T je vektor naměřených dat, e^T je vektor odchylek a $\hat{\Theta}$ je vektor odhadu parametrů.

Matice F má pak tvar:

$$\begin{bmatrix} -y(n) & -y(n-1) & \dots & -y(1) & u(n) & u(n-1) & \dots & u(1) \\ -y(n+1) & -y(n) & \dots & -y(2) & u(n+1) & u(n) & \dots & u(2) \\ \vdots & & & & & & & \vdots \\ -y(N-1) & -y(N-2) & \dots & -y(N-n) & u(N-1) & u(N-2) & \dots & u(N-n) \end{bmatrix} \quad (23)$$

Kde N je počet souborů naměřených vstupních a výstupních dat, n řád diferenční rovnice.

Vektory y a e mají dimenze $(N-n, 1)$ a $\hat{\Theta}(2n, 1)$.

Odhad parametrů modelu je pak:

$$\hat{\Theta} = (F^T F)^{-1} F^T y \quad (24)$$

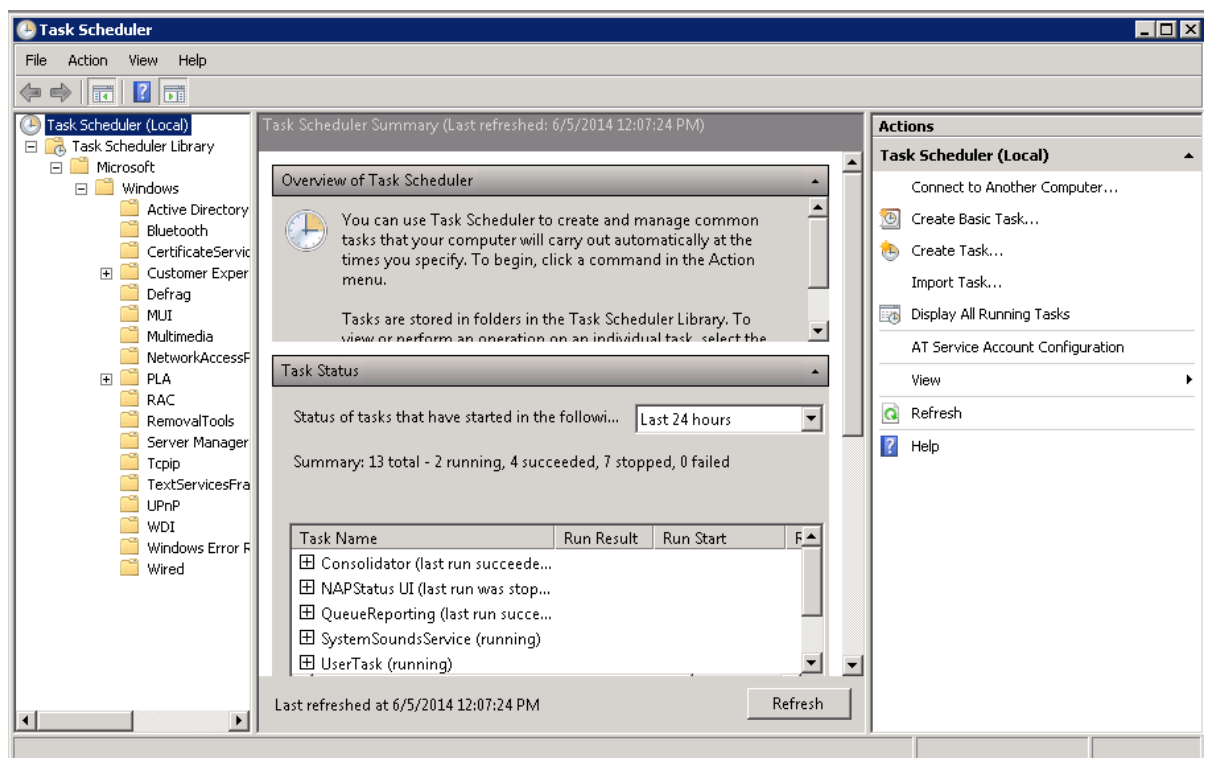
Metoda nejmenších čtverců se často používá jako startovací algoritmus pro složitější a přesnější metody.

[1]

6 AUTOMATICKÉ PLÁNOVÁNÍ ÚLOH VE WINDOWS

Automatické plánování úloh může sloužit k mnoha účelům, ať už k vypnutí počítače, upozornění na nějakou událost, nebo spouštění uživatelských skriptů. Plánovač úloh nabízí celou řadu možností jak úlohy plánovat. k dispozici je i celá řada podmínek za jakých mají být úlohy spuštěny.

Na následujícím obrázku je zobrazen plánovač úloh.



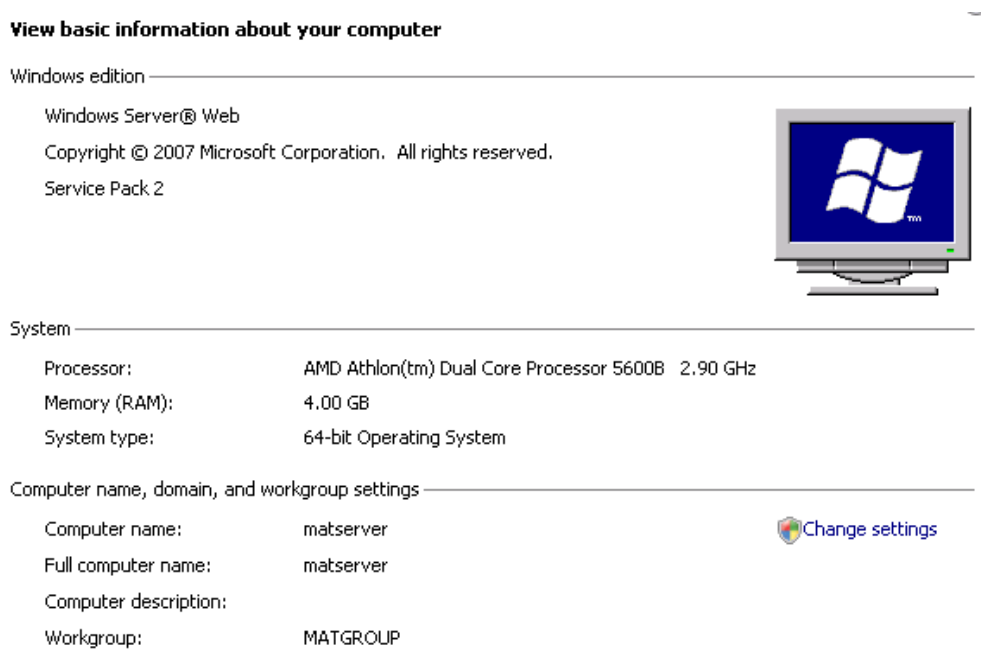
Obrázek 11: Plánovač úloh

II. PRAKTICKÁ ČÁST

7 NASTAVENÍ WEBOVÉHO SERVERU

7.1 Konfigurace webového serveru

Webový server, na kterém běží internetová aplikace pro identifikaci z experimentálních dat, byl použit již existující a nebylo nutné znovu vytvářet nový. Na jednom webovém serveru tak běží současně více internetových aplikací. Název serveru je MATSERVER. Na následujícím obrázku jsou vidět základní informace o serveru:



Obrázek 12: Základní informace o serveru

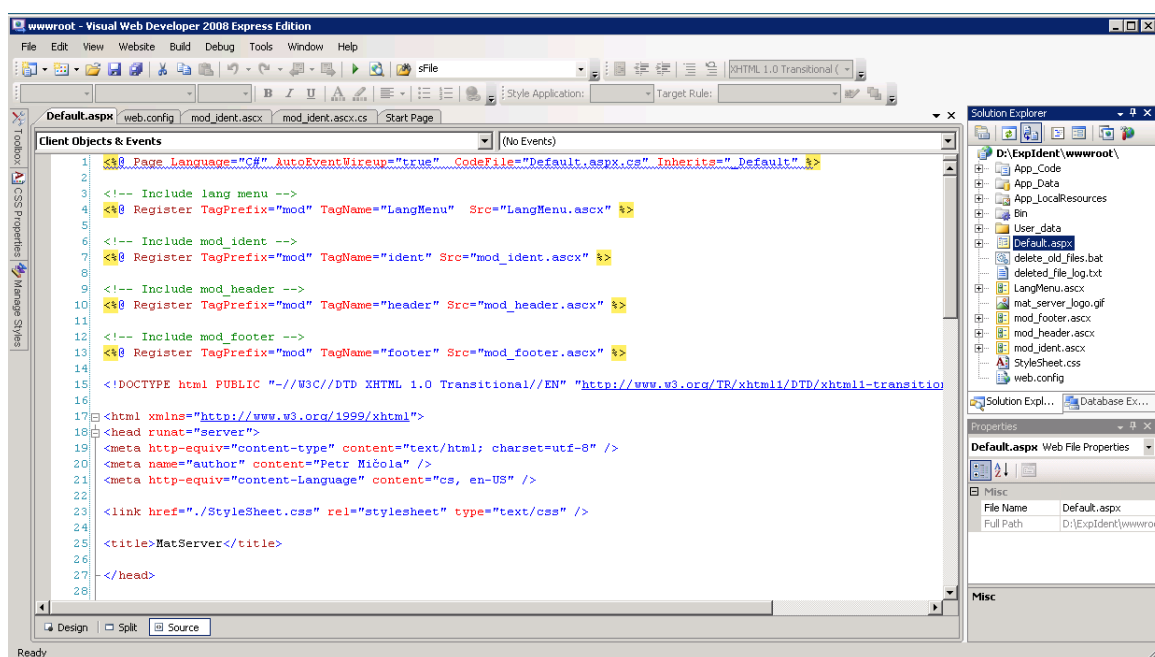
Webový server používá operační systém Windows Web Server 2008 64 bit. Pro potřeby internetové aplikace je plně dostačující, neboť tato distribuce Windows je k tomu určena. Nebyl důvod cokoli na stávající konfiguraci měnit.

Pro připojení na server byl zřízen uživatelský účet s příslušnými právy omezenými na danou oblast působnosti. Pomocí vzdálené plochy bylo navázáno spojení za účelem nahrání aplikace na server a její doladění. Vzdálená plocha je předinstalovaná služba operačního systému Windows 7 Professional, přes který bylo na server přistupováno.

7.2 Vývojové prostředí pro tvorbu internetových aplikací

Na serveru bylo již předinstalované vývojové prostředí *Visual Studio Web Developer 2008 Express Edition* pomocí, pomocí kterého lze upravovat internetovou aplikaci a dále vyvíjet přímo na serveru. Tato verze *Visual Studia* je zdarma jelikož neobsahuje všechny vlastnosti placené verze. Nicméně pro vývoj menších a středních projektů je naprosto dostačující.

[2]



Obrázek 13: Ukázka Visual Studio Web Developer 2008 Express Edition

7.3 MATLAB na serveru

Na webovém serveru byl již dříve nainstalován také MATLAB verze R2008b a R2012b. Tento systém byl využit k programování komponent, který byly poté naimportovány do webové aplikace.

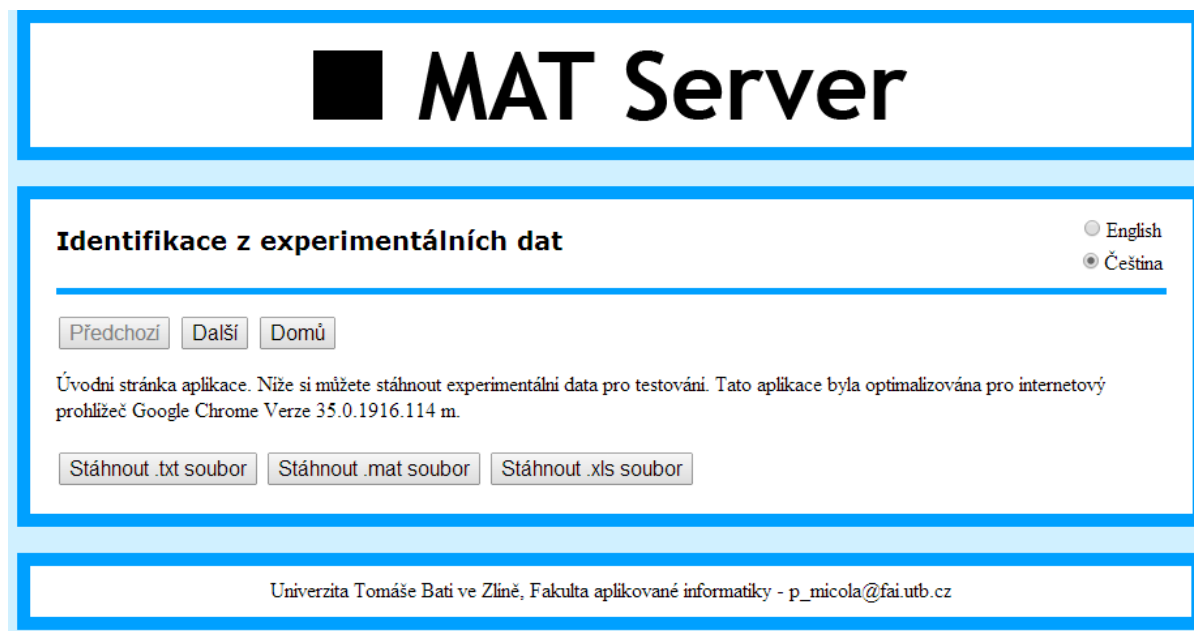
Pokud by bylo podmínkou, aby internetová aplikace běžela na serveru, kde není nainstalován MATLAB, stačilo by mít nainstalovaný MATLAB *Component Runtime*. Avšak na serveru by poté nebylo umožněno spustit plnohodnotný MATLAB a programovat v něm. Běh MATLAB funkcí by byl omezen pouze prostřednictvím zkompileovaných komponent.

8 UŽIVATELSKÉ ROZHRANÍ INTERNETOVÉ APLIKACE

Internetová aplikace pro identifikaci z experimentálních dat je koncipovaná jako dvoujazyčná. Uživatel je krok po kroku provázen aplikací pomocí navigačních tlačítek. Vstupy z hlediska uživatele jsou řešeny pomocí *input* boxů a *radio* boxů. V následujících kapitolách bude stručně popsáno rozložení stránky. V úvodní kapitole bude na obrázku zobrazena i hlavička s patičkou. Další kapitoly kvůli úspoře místa budou obsahovat pouze tělo dané karty. Aplikace je dostupná na <http://matserver.utb.cz/ExpIdent/>. Po pořízení následujících snímků se design aplikace nepatrně změnil. Byly provedeny menší úpravy ve stylování a přibýlo tlačítko do navigace Domů, které slouží pro rychlý návrat na úvodní stránku aplikace.

8.1 Úvod

Úvodní stránka slouží spíše k seznámení. Uživatel na ní má možnost stáhnout si do svého počítače testovací data ve formátu .txt, .mat, .xls. V úvodu je také zmíněno, že aplikace byla navržena a převážně testována pro internetový prohlížeč Google Chrome Verze 35.0.1916.114 m.

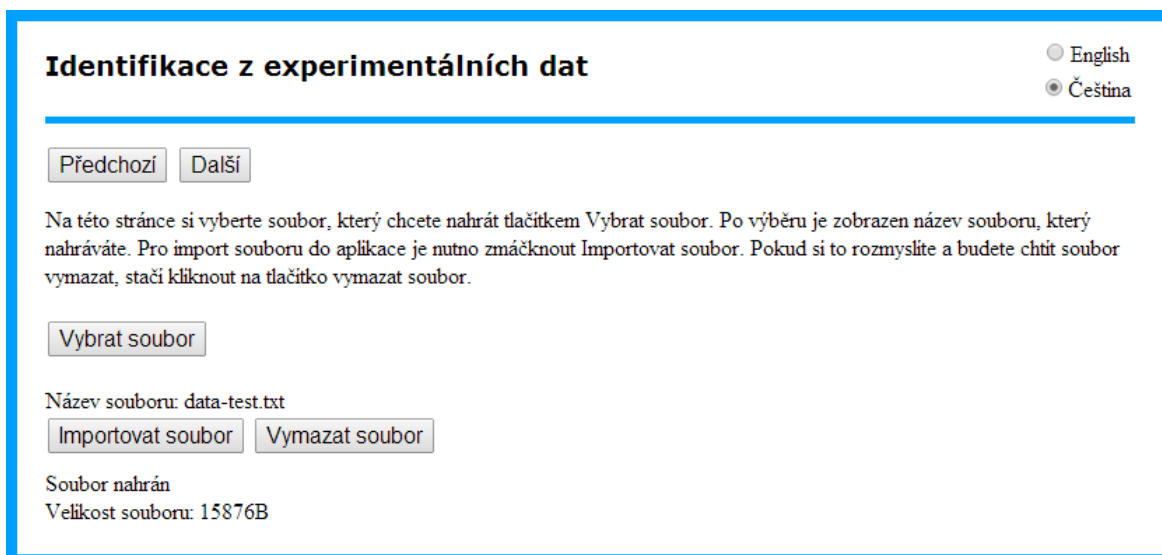


Obrázek 14: Úvodní stránka aplikace

8.2 Výběr a Import souboru

Obrazovka výběr a import souboru slouží k nahrání dat, která mají být identifikována. Uživatel vyhledá a vybere soubor ve svém počítači pomocí vyskakovacího dialogu.

Zobrazí se mu název jeho importovaného souboru. Po stisknutí tlačítka importovat soubor dojde k načtení souboru. Po úspěšném nahrání je uživatel informován textem na obrazovce. Je zobrazena i velikost nahraného souboru. Načtený soubor lze vymazat z paměti pomocí tlačítka Vymazat soubor.



The screenshot shows a web interface titled "Identifikace z experimentálních dat". In the top right corner, there are two radio buttons for language selection: "English" (unselected) and "Čeština" (selected). Below the title, there are two buttons: "Předchozí" and "Další". A paragraph of text explains the process: "Na této stránce si vyberte soubor, který chcete nahrát tlačítkem Vybrat soubor. Po výběru je zobrazen název souboru, který nahráváte. Pro import souboru do aplikace je nutno zmáčknout Importovat soubor. Pokud si to rozmyslíte a budete chtít soubor vymazat, stačí kliknout na tlačítko vymazat soubor." Below this text is a "Vybrat soubor" button. Underneath, the text "Název souboru: data-test.txt" is displayed. Below the name are two buttons: "Importovat soubor" and "Vymazat soubor". At the bottom, it says "Soubor nahrán" and "Velikost souboru: 15876B".

Obrázek 15: Výběr a import souboru

Internetová aplikace počítá pouze s omezenou strukturou dat. Textový soubor musí mít jednotlivá pole oddělená středníkem a na prvním řádku musí být uvedeny názvy jednotlivých sloupců. Soubor typu xls je vyžadován ve tvaru takovém, aby byly na prvním řádku uvedeny názvy sloupců a všechna data na prvním listu sešitu.

8.3 Určení proměnných z importovaného souboru

V dalším kroku je nutné, aby uživatel identifikoval a určil, jaké proměnné odpovídají struktuře jeho souboru. Jména dostupných proměnných jsou vybrána pomocí *dropdown* seznamu. Po načtení proměnných může uživatel vykreslit průběhy nahraných dat.



Identifikace z experimentálních dat English
 Čeština

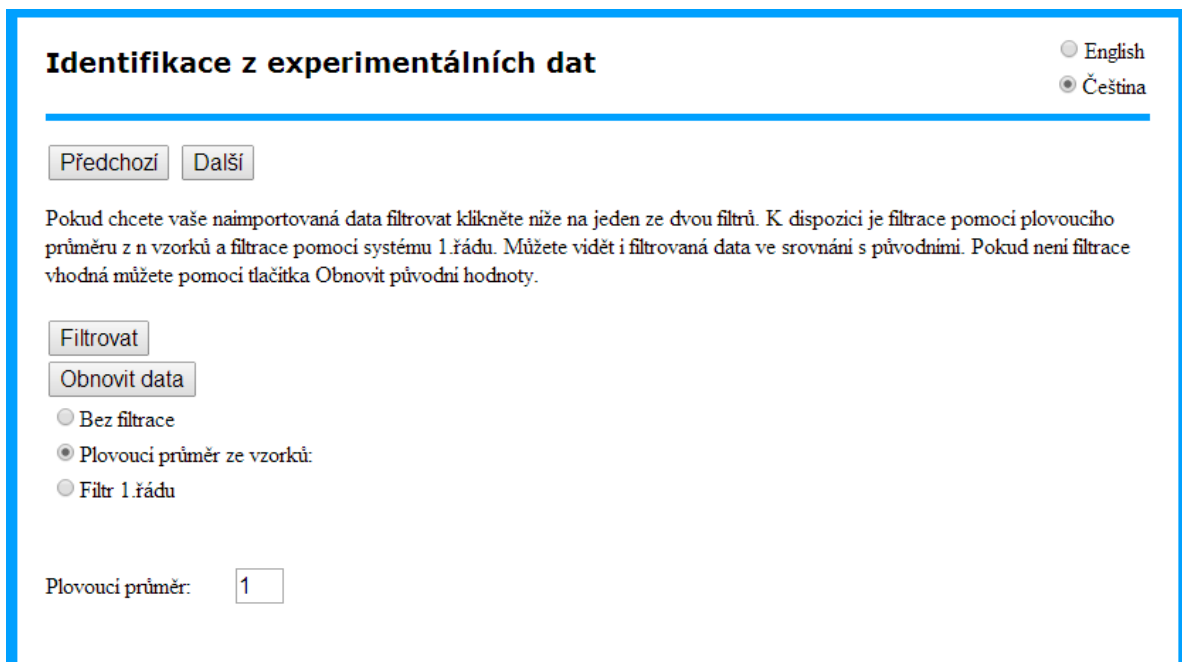
Niže prosím vyberte dle vašeho rozložení v souboru vstupní, výstupní a časovou veličinu ze dropdown listu. Po vybrání klikněte na tlačítko Načíst. Pokud chcete zobrazit průběh nainportované veličiny klikněte na tlačítko Vykreslit.

Vstup u:
Výstup y:
Čas t:

Obrázek 16: Výběr a načtení jednotlivých proměnných ze souboru

8.4 Filtrace importovaných dat

V případě, že uživatel má více zašuměná data, je možné tyto data filtrovat použitím jednoduchého filtru. k dispozici je filtr plovoucí průměr z n vzorků a filtrace pomocí soustavy 1.řádu se zvolenou časovou konstantou. Uživatel zadá do příslušného pole danou hodnotu a po stisknutí tlačítka filtrovat jsou vykreslena původní a filtrovaná data. Uživatel má možnost zrušit filtrování a použít původní data pokud to uzná za vhodné.



Identifikace z experimentálních dat English
 Čeština

Pokud chcete vaše nainportovaná data filtrovat klikněte níže na jeden ze dvou filtrů. K dispozici je filtrace pomocí plovoucího průměru z n vzorků a filtrace pomocí systému 1.řádu. Můžete vidět i filtrovaná data ve srovnání s původními. Pokud není filtrace vhodná můžete pomocí tlačítka Obnovit původní hodnoty.

Bez filtrace
 Plovoucí průměr ze vzorků:
 Filtr 1.řádu

Plovoucí průměr:

Obrázek 17: Filtrace uživatelských dat

8.5 Zvolení stupňů polynomů ARX modelu a výpočet

Na této kartě uživatel zvolí, jakou soustavou chce identifikovat vložená data. Do příslušných polí zadá stupně polynomů čitatele a jmenovatele ARX modelu. Po vypočítání a identifikaci dojde k vykreslení Z-přenosu aproximované soustavy a vybraných statistických ukazatelů.

Identifikace z experimentálních dat

English
 Čeština

V tomto kroku vyberte stupně polynomu modelu jimž chcete aproximovat data. Po té klikněte na tlačítko Vypočítat a zobrazí se aproximovaný model.

Stupeň čitatele:

Stupeň jmenovatele:

Obrázek 18: Karta pro zadání stupňů polynomů ARX modelu

V tomto kroku vyberte stupeň polynomu modelu jímž chcete aproximovat data. Po té klikněte na tlačítko Vypočítat a zobrazí se aproximovaný model.

Stupeň čitatele:

Stupeň jmenovatele:

$$\frac{b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

b [0 0.2630 0.1519]
a [1.0000 -1.6741 0.7578]

Reziduální chybová funkce 1.5563
Rozptyl: 0.0031
Chyba predikce: -0.0381
Kovarianční matice :
0.0501 -0.0479 -0.0023 0.0122
-0.0479 0.0499 0.0021 -0.0120
-0.0023 0.0021 0.0203 -0.0012
0.0122 -0.0120 -0.0012 0.0232

Obrázek 19: Model identifikované soustavy – testovací data

8.6 Srovnání identifikovaného modelu s uživatelskými daty

V posledním kroku je možné zobrazit spojitý model aproximované soustavy. Dále je zobrazeno porovnání průběhu výstupu identifikovaného modelu s reálnými výstupními uživatelskými daty.

Identifikace z experimentálních dat

English
 Čeština

[Předchozí](#) [Další](#)

Tato stránka slouží k zobrazení spojitého aproximovaného přenosu. Dále je v grafu porovnán průběh modelované výstupní veličiny a reálně naměřená data. Pro zobrazení grafu klikněte na tlačítko Zobrazit spojitý model a srovnání modelu a reálných dat.

[Zobrazit spojitý model a srovnání modelu a reálných dat](#)

[Export výsledků .mat](#)

Obrázek 20: Karta pro zobrazení spojitého modelu

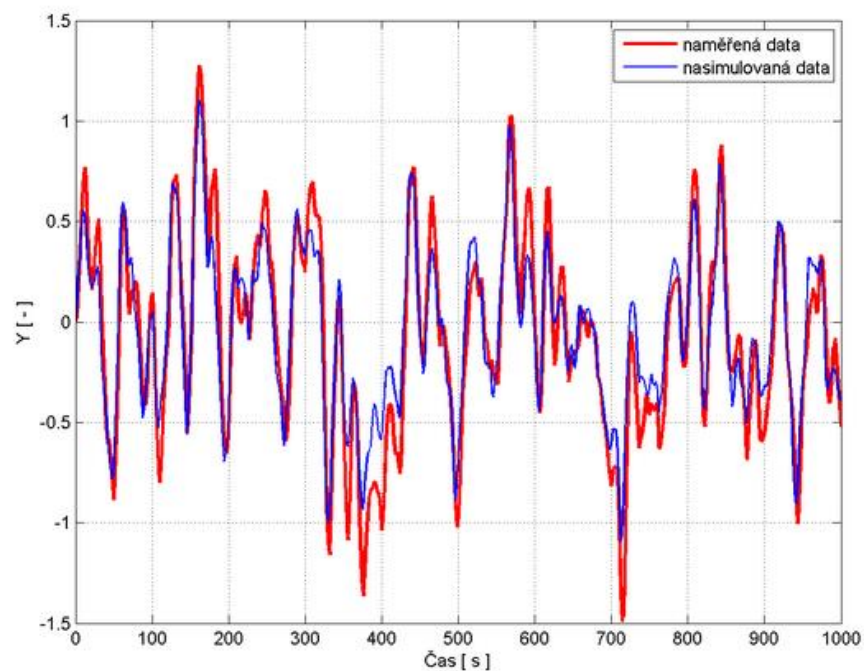
Zobrazit spojitý model a srovnání modelu a reálných dat

$$b_2*s^2 + b_1*s^1 + b_0$$

$$a_2*s^2 + a_1*s^1 + a_0$$

$$b [0 \ 0.0267 \ 0.1197]$$

$$a [1.0000 \ 0.1387 \ 0.0242]$$

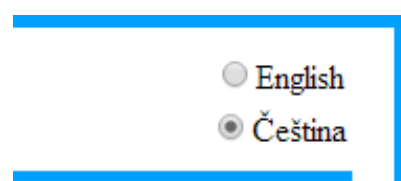


Export výsledků .mat

Obrázek 21: Spojitý model a porovnání průběhu identifikovaného modelu a reálných dat

8.7 Změna jazyka

V pravém horním rohu je uživateli nabídnuta volba jazyka. Jednoduchým kliknutím, tak uživatel může změnit lokalizaci aplikace.



Obrázek 22: Přepínač jazyka

8.8 Uživatelské testování

Nejlepším testem funkčnosti a přehlednosti internetové aplikace je uživatelské testování. Vybraní kolegové testovali aplikaci a její uživatelské rozhraní. Cílem uživatelského testování bylo dojít na konec aplikace bez zásahu pozorovatele. Všichni uživatelé bez problémů zobrazili cílovou stránku a neměli velké výhrady ke vzhledu. Naopak ocenili čistý design a přehlednost aplikace. Během testování nebyly zjištěny žádné závažné chyby ve vzhledu.

Při vytvoření aplikace na serveru se občas vyskytnuly problémy, kdy aplikace zobrazovala výjimky. Zobrazované výjimky byly prozkoumány, avšak příčinu se nepodařilo odhalit. Tyto stavy se již nepodařilo dále nasimulovat a bez zásahu vývojáře se již nezobrazovaly.

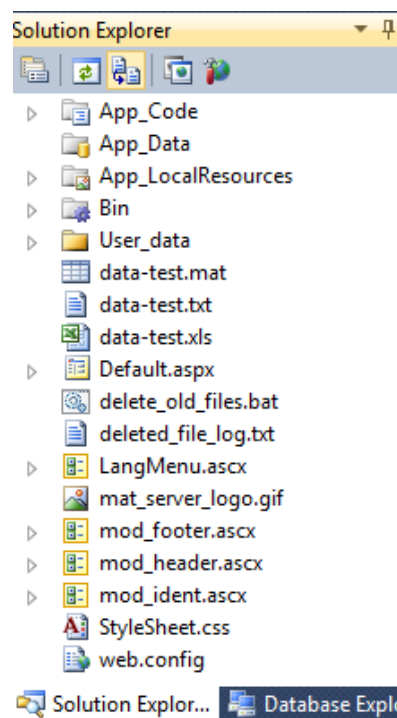
9 KÓDOVÉ POZADÍ VYTVOŘENÉ APLIKACE

V této kapitole bude stručně popsáno kódové pozadí vytvořené aplikace. Budou uvedeny pouze příklady, neboť samotný kód aplikace je velmi rozsáhlý a přesahuje tisíc řádků. U internetových aplikací je velmi důležitá dokumentace. Uživatelská dokumentace je stručně uvedena v předcházející kapitole. Tato část tedy bude spíše takovou vývojářskou dokumentací. Nejlepší dokumentací je samotný kód, který má být snadno čitelný a samodokumentovatelný. i přes tento fakt jsou v programovém kódu uvedeny komentáře k jednotlivým blokům pro snazší správu aplikace.

Vzhledem k problému při kompilaci ve Visual studiu na straně serveru byla aplikace odladěna na lokálním stroji. Výsledné kódy potom byly nahrány na server a aplikace spuštěna. Programování pozadí jednotlivých komponent a funkcí bylo prováděno v jazyce C#.

9.1 Struktura internetové aplikace

Stručný popis struktury aplikace již byl popsán v kapitole 2. Nyní budou popsány přímo jednotlivé složky a soubory k čemu slouží.



Obrázek 23: Struktura aplikace

Složka App_Code slouží k uložení zvláštních tříd, které si vývojář naprogramuje a jejich funkcí pak využívá. V této aplikaci je zde uložen soubor BasePage.cs, který reprezentuje

třidu BasePage. Z této třídy je odvozena Default.aspx.cs. Třída BasePage slouží především k lokalizaci aplikace.

Složka App_Data je určena soubory, které využívá lokální databáze. V této internetové aplikaci nejsou využity žádné databáze a tak je tato složka prázdná.

Složka App_LocalResources je určena pro soubory, které slouží k lokalizaci aplikace. V těchto souborech jsou uloženy překlady k jednotlivým prvkům v aplikaci. Podrobnější popis této složky bude v kapitole o lokalizaci aplikace.

Složka Bin je dedikovaná pro uložení .NET knihoven. V této aplikaci obsahuje knihovny matLib.dll, matLibNative.dll, MWArray.dll a WebFiguresService.dll.

Do složky User_data se ukládají uživatelské soubory. Jedná se o .mat, .xls, .txt soubory, vykreslené grafy ve formě .png obrázků a exportované .mat soubory.

Soubory data-test.mat, data-test.txt a data.xls slouží k poskytnutí testovacích dat uživateli.

Soubor Default.aspx je první stránkou, která se uživateli načte. Do této stránky jsou nalinkovány všechny ostatní moduly. Programové pozadí je uloženo v souboru Default.aspx.cs.

Soubor delete_old_files.bat je skript určený pro mazání starých souborů ve složce User_data a bude popsán později.

Soubor deleted_file_log.txt je textový soubor, který slouží jako záznam. Jsou v něm uloženy názvy všech smazaných souborů pomocí skriptu delete_old_files.bat.

Soubor LangMenu.ascx je modul, který obsahuje pouze přepínač na změnu jazyka. Jeho datové pozadí je uloženo v souboru LangMenu.ascx.cs.

Soubor mat_server_logo.gif je obrázek, který je načítán do modulu hlavičky. Tento obrázek byl převzat z matserver.utb.cz uvedeném ve zdroji [10].

Soubor mod_footer.ascx je webový modul, který slouží k zobrazení patičky v internetové aplikaci. Jeho programové pozadí je uloženo v souboru mod_footer.ascx.cs.

Soubor mod_header.ascx je podobně jako předchozí webový modul. Tento ovšem slouží k zobrazení informací v patičce internetového dokumentu. Datovému pozadí je určen soubor mod_header.ascx.cs.

Soubor `mod_ident.ascx` je webový modul. Tento modul má největší rozsah a je jádrem této internetové aplikace. Obsahuje hlavní tělo, pomocí kterého jsou řízeny uživateli „kroky“. Programové pozadí je v souboru `mod_ident.ascx.cs`.

Soubor `StyleSheet.css` je určen pro externí uložení nadefinovaných CSS stylů

Soubor `web.config` je konfigurační soubor webového serveru. Je generován automaticky, ale je možné ho upravovat i ručně.

9.2 M-soubory použité pro vytvoření komponenty

V této podkapitole budou stručně popsány funkce jednotlivých m-souborů, zkompilování m-souborů do knihovny a jejich volání z objektu vytvořeného v C#.

9.2.1 Soubory `mFilterN.m`, `mFilterP.m`

Tyto funkce slouží pro filtraci vstupních dat pomocí vestavěné funkce `filter` z programového systému MATLAB. Funkce jsou naprogramované tak, aby vrátily `MWNumericArray`. Tyto hodnoty přepíší původní načtená data od uživatele (výstupní veličinu `y`).

Nejdříve je vytvořen objekt dané třídy. Pak je funkce zavolána stejně jako každá jiná metoda objektu. Příklad volání funkce `mFilterN.m`:

```
matLoadClass mFilter = new matLoadClass();  
mwnaY = (MWNumericArray)mFilter.mFilterN(mwnaY,  
Convert.ToDouble(tbFilterValueN.Text));
```

9.2.2 Soubory `mLoad.m`, `mLoadTXT.m`, `mLoadXLS.m`

Tyto funkce slouží k načtení vstupních dat od uživatele. Rozděleny jsou podle toho, který typ vstupního souboru uživatel importuje (`.mat`, `.txt`, `.xls`).

Cílem bylo naprogramovat všechny tyto funkce takovým způsobem, aby vracely stejnou výstupní formu. Tímto bylo zajištěno, že nemusel být přepsán zbytek následujícího kódu, což by bylo značně nežádoucí a nepraktické. Typ návratové proměnné je `MWStructArray`.

Volání jednotlivých funkcí zajišťuje přepínač, který podle přípony vstupního souboru od uživatele rozhoduje, která ze zmíněných funkcí bude zavolána a použita.

Přepínač a volání žádoucích funkcí:

```
switch (sExtension)
{
    case ".mat":
        mwsaMtlAr = (MWStructArray)mLoader.mLoad(sFileName);
        break;
    case ".txt":
        mwsaMtlAr = (MWStructArray)mLoader.mLoadTXT(sFileName);
        break;
    case ".xls":
        mwsaMtlAr = (MWStructArray)mLoader.mLoadXLS(sFileName);
        break;
}
```

9.2.3 Soubory mPlot.m, mPlotMulti.m

Pro zobrazení a vykreslení výsledků ve formě grafu byly naprogramovány funkce mPlot.m, mPlotMulti.m. Tyto funkce využívají standardní vestavěnou funkci plot ze systému MATLAB. Možnosti této funkce jsou opravdu velké a je jen na vývojáři, které z těchto funkcí zpřístupní uživateli. Tato funkce nevrací žádnou hodnotu, respektive její návratový typ je void. Výsledný soubor s grafem se ukládá ve formě png obrázku.

9.2.4 Soubory sD2C.m, sDen.m, sNum.m

Byla vytvořena funkce sD2C.m. Tato funkce slouží k přepočtu zadaného vstupního diskrétního přenosu na jeho spojitou verzi. Využívá se vestavěné funkce d2c. Návratovou hodnotou je MWNumericArray. Ve výstupní proměnné jsou uloženy hodnoty koeficientů čitatele a jmenovatele. Pro získání těchto koeficientů byly vytvořeny funkce sDen.m a sNum.m, které pouze vrátí ze vstupní proměnné danou část.

Jedním z parametrů sD2C.m je název souboru. Do tohoto souboru se uloží veškeré výpočty provedené v MATLABu při této funkci. Uloží se tzv. workspace.

9.2.5 Soubor sSim.m

Pro simulaci odezvy systému na zadaný vstup byla naprogramována funkce sSim.m. Na základě identifikovaných polynomů přenosu a vektoru akční veličiny je vypočítána odezva systému. Využito je funkce lsim. Návratová hodnota je MWStructArray. Vrací vektor výstupní a časové veličiny. Výstup je v programovém pozadí aplikace uložen do listu a z něj se pak volají pomocí názvů položek ve struktuře jednotlivé vektory.

Příklad uložení a zpracování vrácené MWStructArray proměnné:

```
matCalculateClass mSimulateModel = new matCalculateClass();
```

```
MWStructArray mwsaModelSimulation = null;
mwsaModelSimulation = (MWStructArray)mSimulateModel.sSim(mwnaU, mwnaT, b,
a);
Dictionary<String, MWArray> dMatArraySim = new Dictionary<string,
MWArray>();
List<String> lMatArraySim = mwsaModelSimulation.FieldNames.ToList();
foreach (String sMatVariable in lMatArraySim)
{
    dMatArraySim.Add(sMatVariable.ToString(),
mwsaModelSimulation.GetField(sMatVariable)); }

    mwnaTModelSim = (MWNumericArray) dMatArraySim["t"];
    mwnaYModelSim = (MWNumericArray) dMatArraySim["y"];
```

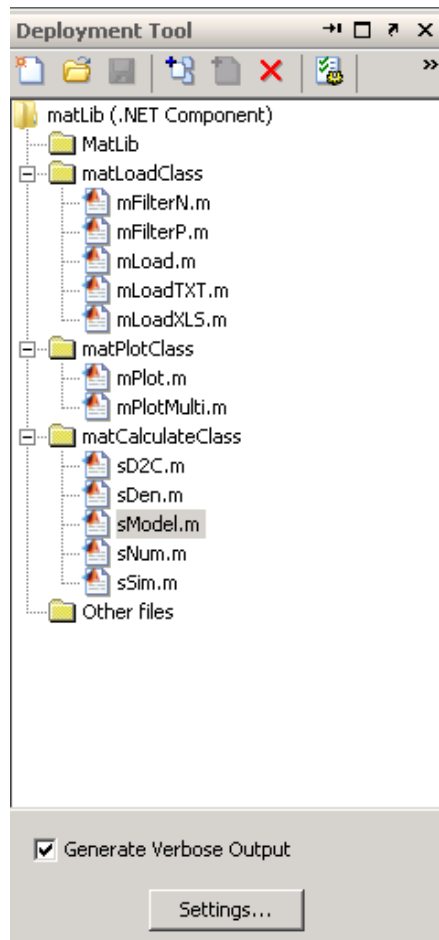
9.2.6 Soubor sModel.m

Pro identifikaci uživatelem zadaných dat byla vytvořena funkce sModel.m. Tato funkce obsahuje algoritmus, který pomocí metody nejmenších čtverců vrátí parametry modelu. Návrátová hodnota je MWStructArray, kde je uložen vektor koeficientů jmenovatele, čitatele, rozptyl, residuální chybová funkce, chyba predikce, kovarianční matice odhadu parametrů.

Výstup je uložen do zadaného .mat souboru. Ukládá se celý pracovní prostor MATLABu.

9.2.7 Export naprogramovaných m-souborů do dll knihoven

Všechny vytvořené m-soubory bylo nutné zkompileovat pomocí MATLAB Builder NE a následně pak naimportovat do vlastní C# aplikace. Kompilace v programu MATLAB byla provedena za pomoci tzv. deployment tool. Nejdříve byl vytvořen projekt typu .NET Component s názvem matLib. Následně byly v této komponentě vytvořeny jednotlivé třídy. V deployment tool se jako třídy berou jednotlivé složky. Metody tříd se vytvářejí přidáním souborů do těchto složek. Byly tedy do každé složky přidány odpovídající m-soubory. Následně byla spuštěna kompilace.



Obrázek 24: Deployment tool

Po dokončení kompilace byly vytvořeny soubory `matLib.dll` a `matLibNative.dll`. Tyto soubory byly přidány do složky `Bin` internetové aplikace. Následně byla vytvořena reference na tyto knihovny v prostředí Visual Studia.

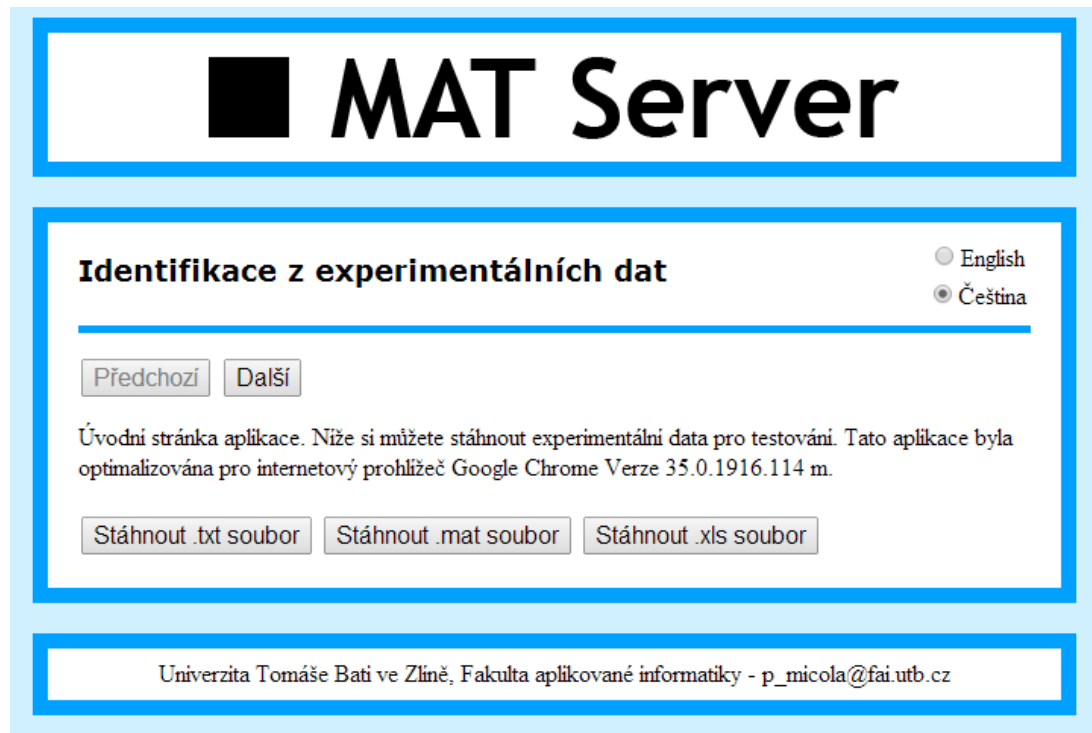
Díky tomuto postupu bylo možné používat naprogramované funkce z prostředí MATLAB v internetové aplikaci.

9.3 Soubory CS, ASPX, ASCX

V této části budou stručně popsány jednotlivé soubory z pohledu vývojáře. Podrobnější popis je přímo v kódu daných funkcí.

9.3.1 Soubor `Default.aspx`

Tato stránka je jako první zobrazena uživateli. Do této stránky se formou webových modulů přidávají další komponenty. Výsledkem pak byla celá internetová aplikace. Výhodou modulů je jednoduchý import do stránky i případné odebrání.



Obrázek 25: Úvodní stránka internetové aplikace

Stránka je rozdělena na základní bloky hlavičku, jazykové menu, tělo a patičku. Tyto moduly je nutné do stránky tzv. zaregistrovat.

Příklad zaregistrování modulu patičky vypadá následovně:

```
<%@ Register TagPrefix="mod" TagName="footer" Src="mod_footer.ascx" %>
```

Nyní již bylo možné použít daný modul přímo v těle stránky. Volání modulu patičky v těle stránky:

```
<mod:footer id="mod_footer" runat="server" />
```

9.3.2 Soubor mod_ident.ascx

Tato stránka je webový modul, který byl naimportován do úvodní stránky aplikace. V tomto modulu je řešeno vše podstatné a mohl by fungovat samostatně za předpokladu, že by nebyla vyžadována lokalizace. V takovém případě by nemusely být načteny moduly s lokalizací.

Nyní bude stručně popsáno několik základních ASP.NET prvků využívaných v aplikaci. Jak již bylo zmíněno dříve, uživatel se pohybuje aplikací pomocí tlačítek Další a Předchozí. Obdobným způsobem jako tyto jsou vytvořena i další tlačítka použitá

v aplikaci. Rozdíl je pouze v jejich vlastnostech. Při stisku tlačítka se volá funkce přiřazená vlastnosti `onclick`. Tato funkce je naprogramována v `cs` souboru.

Tlačítko Další je definováno například následovně:

```
<asp:Button id="btnNext" runat="server" Text="" onclick="btnNextClick" />
```

Dalším velmi často používaným prvkem byl popis textu. Použit byl i při lokalizaci jednotlivých textů a oznámení uživateli. Příklad definice v ASP.NET:

```
<asp:Label id="lblIntroInfo" runat="server" Text="Úvodní stránka">
</asp:Label>
```

Stránky byly uživateli zobrazovány pomocí prvku `MultiView`. Tento prvek vrací hodnotu aktuálního indexu na stránce. Pomocí tohoto indexu se pak uživateli zobrazovaly požadované dílčí stránky, které byly uloženy v jednotlivých částech `MultiView`. Tyto části jsou implementovány prvky `View`. Implementace vypadala následovně:

```
<asp:MultiView id="mvBody" runat="server">
    <!-- View 0 Description -->
    <asp:View id="View0" runat="server">
        </asp:View>
</asp:MultiView>
```

Pro načtení souboru od uživatele bylo využito ASP.NET prvku `FileUpload`. Tento prvek má řadu naprogramovaných funkcí, které vývojáři usnadňují práci s importovaným souborem. Vstup uživatele byl testován pomocí ASP.NET prvku `RegularExpressionValidator`. Povoleny jsou pouze soubory s koncovkou `mat`, `xls`, `txt`. Základní implementace v ASP.NET může vypadat následovně:

```
<asp:FileUpload id="fuPopupFile" runat="server" />
<asp:RegularExpressionValidator id="revFileUpload" runat="server"
ControlToValidate="fuPopupFile"
ValidationExpression="(.\.([Mm][Aa][Tt])|.\.([Xx][Ll][Ss])|.\.([Tt][Xx]
][Tt]))"></asp:RegularExpressionValidator>
```

Pro přiřazení vstupních dat do jednotlivých proměnných byl využit prvek `DropDownList`. Obsah je dynamicky plněn z pozadí aplikace a uživatel si ze seznamu vybere, která hodnota reprezentuje vstupní, výstupní a časovou veličinu. Programový kód může vypadat následovně:

```
<asp:DropDownList id="ddlT" runat="server"></asp:DropDownList>
```

Zobrazení grafů uživateli je řešeno pomocí prvku `Image`. Tento prvek je obalen prvkem typu `HyperLink`. Tato konstrukce pak umožní při kliknutí na graf otevřít nebo stáhnout daný obrázek v plném rozlišení. Implementace v ASP.NET vypadá následovně:

```
<asp:HyperLink id="hlNewData" runat="server" Target="_blank">
  <asp:Image id="iNewData" runat="server" />
</asp:HyperLink>
```

9.4 Lokalizace aplikace

Za účelem dvoujazyčnosti aplikace byla řešena lokalizace aplikace do anglického a českého jazyka. Defaultním jazykem je zvolena čeština.

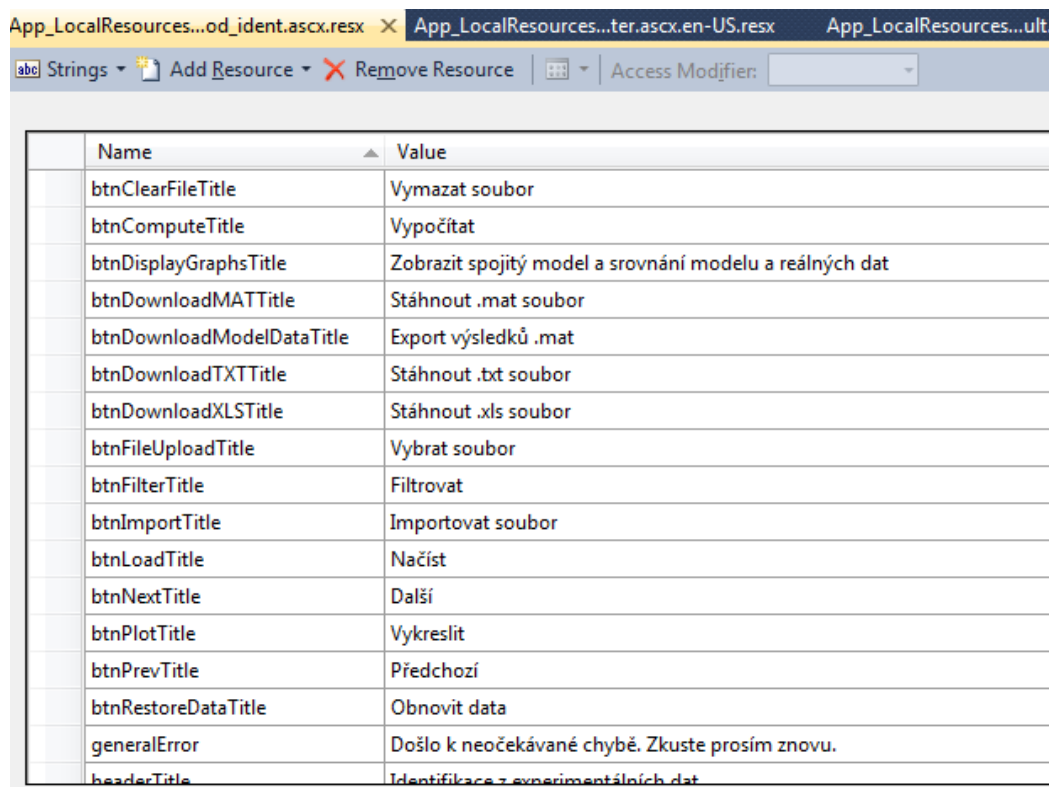
Základní přepínání jazyků je řešeno v modulu `LangMenu.ascx`. Pomocí ASP.NET prvku `RadioButtonList` je získán požadovaný jazyk a ten pak uživateli nastaven. Jazyk je uložen ve formě locale do cookies internetového prohlížeče.

```
1 <%@ Control Language="C#" AutoEventWireup="true" CodeFile="LangMenu.ascx.cs" Inherits="LangMenu" %>
2
3 <asp:RadioButtonList ID="rblLangMenu" runat="server" AutoPostBack="True" onselectedindexchanged="rblOnChangeLang">
4   <asp:ListItem Value="en-US">English</asp:ListItem>
5   <asp:ListItem Selected="True" Value="cs-CZ">Čeština</asp:ListItem>
6 </asp:RadioButtonList>
7
```

Obrázek 26: Seznam pro výběr jazyka v souboru `LangMenu.ascx`

Lokalizace jednotlivých prvků na stránce většinou proběhla pomocí jejich vlastnosti `Text`. Při předkreslení stránky se zavolala metoda `GetLocalResourObject("name")`. Tato funkce vrací hodnotu klíče `name` uloženou v příslušném souboru.

Překlady jsou tedy uloženy v složce `App_LocalResources`. Zde jsou zdrojové soubor `*.resx` a `*.en-US.resx`. Texty jsou uloženy ve tvaru `název` a `hodnota`. Na následujícím obrázku je vidět ukázka struktury takového souboru.



Name	Value
btnClearFileTitle	Vymazat soubor
btnComputeTitle	Vypočítat
btnDisplayGraphsTitle	Zobrazit spojitý model a srovnání modelu a reálných dat
btnDownloadMATTitle	Stáhnout .mat soubor
btnDownloadModelDataTitle	Export výsledků .mat
btnDownloadTXTTitle	Stáhnout .txt soubor
btnDownloadXLSTitle	Stáhnout .xls soubor
btnFileUploadTitle	Vybrat soubor
btnFilterTitle	Filtrovat
btnImportTitle	Importovat soubor
btnLoadTitle	Načíst
btnNextTitle	Další
btnPlotTitle	Vykreslit
btnPrevTitle	Předchozí
btnRestoreDataTitle	Obnovit data
generalError	Došlo k neočekávané chybě. Zkuste prosím znovu.
headerTitle	Identifikace z experimentálních dat

Obrázek 27: Ukázka struktury lokální zdrojových souborů pro multijazyčnost

Velkou výhodou takového přístupu je skutečnost, že uživatele nemusí zajímat programový kód aplikace. Překlady může měnit přímo v tomto souboru.

Byly vytvořeny zdrojové soubory pro jednotlivé jazyky a pro jednotlivé stránky. Tímto je zajištěno bezproblémové přidávání nových jazyků do aplikace.

V modulu `mod_ident.ascx` je využito ASP.NET prvku `RangeValidator`. Tento prvek kontroluje vstupy od uživatele. Pokud by zmíněný prvek nebyl použit v řešené aplikaci, byla by předchozí lokalizace dostačující a nebylo by nutné ji dále řešit. Největším problémem byl desetinný oddělovač a jeho rozdílnost v závislosti na použitém jazyce. Pomocí třídy `BasePage` byl tento problém vyřešen. V této třídě je nastavena tzv. `Culture` aplikace na příslušný jazyk. Díky tomuto přístupu identifikuje prvek `RangeValidator`, zda má očekávat tečku nebo čárku. V řešené aplikaci byl problémem pouze desetinný oddělovač. Tento přístup by však vyřešil i všechny ostatní jazykové odlišnosti například formát data.

9.5 CSS internetové aplikace

Vzhled internetové aplikace je formátovaný pomocí externích stylů uložených v souboru `StyleSheet.css`. Jednotlivé moduly určené pro základní rozdělení na hlavičku,

tělo a patičku mají vytvořeny odpovídající třídy. Tímto bylo dosaženo základního rozdělení stránky.

Jednotlivé elementy pak byly nastýlovány pomocí tříd nebo pomocí přístupu přes jejich id. Styly jsou připraveny pro další rozšíření v případě, že by stávající vzhled nedostačoval. Byla snaha jednotlivým větším blokům přidat svou třídu a přes ni formátovat vzhled těchto bloků.

Detailnější popis CSS stylů není třeba. Nejčastěji bylo využito pozicování, ohraničení, šířky, formátování textu.

Styly byly naimportovány do stránky Default.aspx pomocí této direktivy:

```
<link href="./StyleSheet.css" rel="stylesheet" type="text/css" />
```

9.6 Ověření uživatelských vstupů

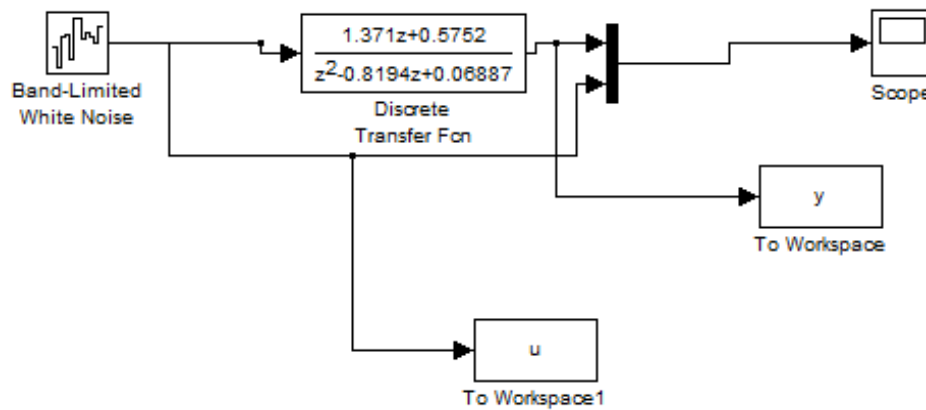
Aplikace povoluje vložit uživateli pouze ty data, která odpovídají validačním kritériím. V řešené aplikaci jsou pro tyto případy využity prvky ASP.NET , a to `RegularExpressionValidator` a `RangeValidator`. Validace chrání aplikaci před vložením nesprávných dat. Řešená aplikace nepoužívá databázi a z tohoto důvodu nebyl tento bod nijak ošetřen. Pro striktní zabezpečení lze nastavit na IIS filtr na omezení přístupu pouze ze známých IP.

10 SROVNÁNÍ VÝSLEDKŮ APLIKACE S VÝSLEDKY Z MATLABU

Exportovaná testovací data z řešené aplikace byly porovnány s výsledky ze systému MATLAB. k porovnání byl využit příkaz `arx` z identifikačního toolboxu. Z následujících kapitol je zřejmé, že výsledky obou přístupů jsou srovnatelné.

10.1 Generování testovacích dat

Testovací data dostupná v řešené internetové aplikaci byly získány simulací z programu Simulink. Dynamická soustava byla buzena náhodným signálem. Hodnoty budícího signálu a výstupní veličiny byly zaznamenány a použity pro testovací účely.



Obrázek 28: Simulační schéma z prostředí Simulink

10.2 Porovnání výsledných diskretních přenosů

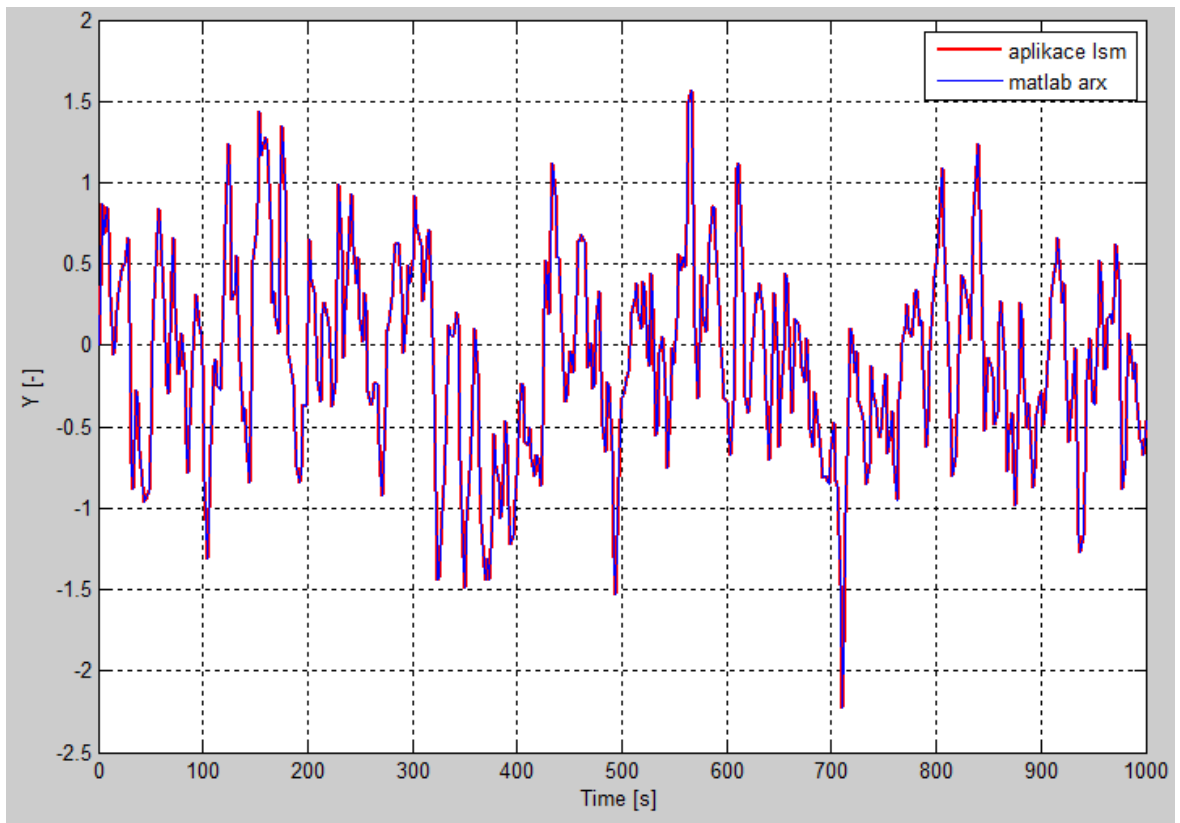
Přenos získaný z internetové aplikace:

$$G(z) = \frac{1,371z^{-1} + 0,5752z^{-2}}{1 - 0,8194z^{-1} + 0,0689z^{-1}} \quad (25)$$

Přenos získaný z programového prostředí MATLAB pomocí příkazu `arx`:

$$G(z) = \frac{1,371z^{-1} + 0,5752z^{-2}}{1 - 0,8194z^{-1} + 0,0689z^{-1}} \quad (26)$$

10.3 Grafické porovnání průběhů



Obrázek 29: Chování identifikovaného modelu z řešené aplikace v porovnání s výstupem funkce arx

11 AUTOMATICKÉ MAZÁNÍ STARÝCH UŽIVATELSKÝCH DAT

Veškerá data uživatele, který používá aplikaci, jsou nahrávána do složky User_data. Pokud by data nebyla průběžně mazána, mohlo by dojít k zaplnění místa, které bylo vyhrazeno pro aplikaci.

11.1 Skript pro mazání starých souborů

Byl naprogramován skript delete_old_files.bat využívající příkazy příkazového řádku. Následně byl tento skript zaplánován jako automatická úloha prostřednictvím plánovače ve Windows. Názvy smazaných souborů se zapisují do vytvořeného logu deleted_file_log.txt.

Naprogramovaný skript delete_old_files.bat:

```
@ECHO OFF

:: AUTHOR: MICOLA PETR

:: DATE: 10-MAY-2014

forfiles /P "D:\ExpIdent\wwwroot\User_data" /S /M *.* /D -7 /C "cmd /c
del @path && ECHO deleting file: @path" >>
D:\ExpIdent\wwwroot\deleted_file_log.txt

EXIT
```

Příkaz forfiles slouží k vykonání zadaných příkazů nad souborem nebo více soubory. Využívá se zejména pro dávkové zpracování. Popis použitých parametrů v příkazu forfiles je následující:

/P – cesta pro vyhledání

/S – rekurzivně i v podsložkách

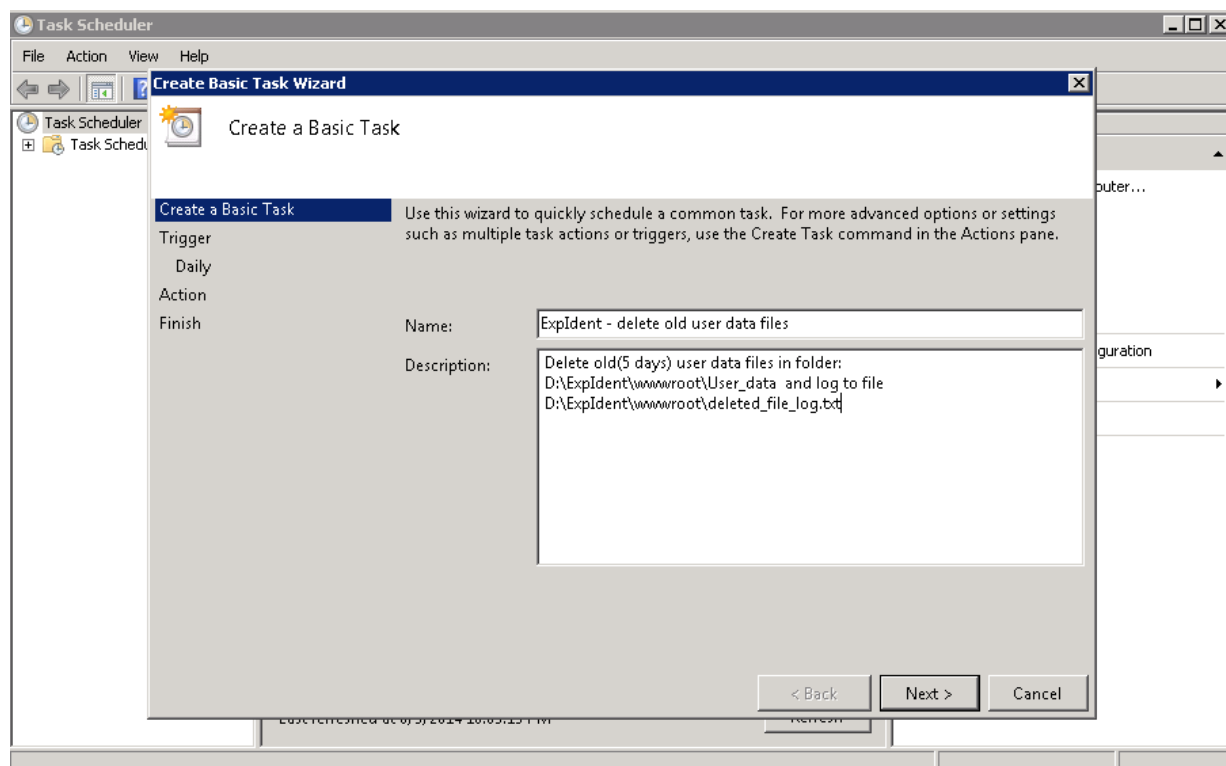
/M – slouží pro výběr souborů, které odpovídají zadané masce

/D – vybere soubory s poslední změnou menších než (-) zadaný počet dní

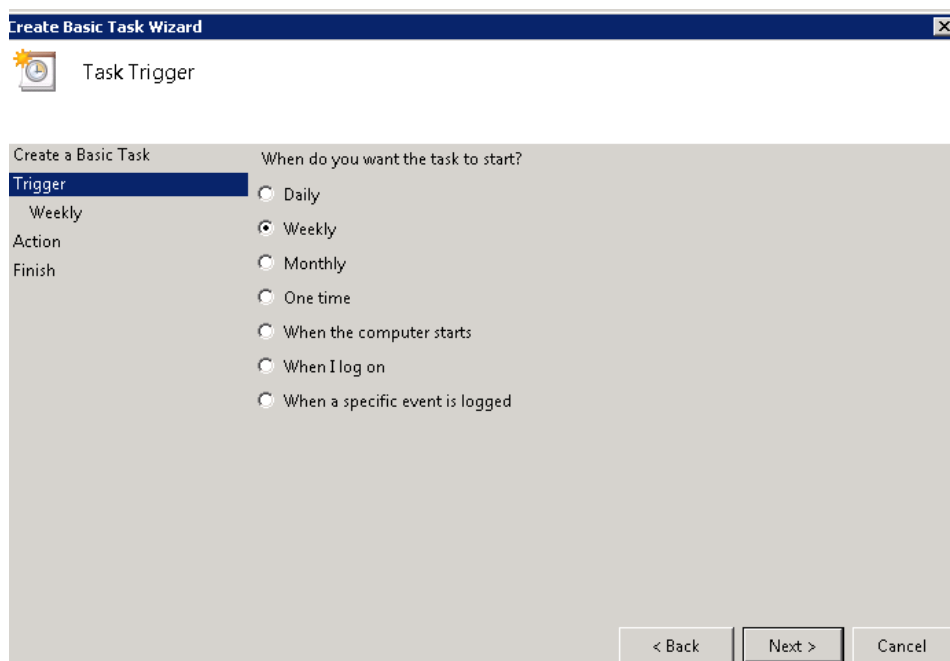
cmd – provede příkaz, v tomto případě smaže soubor se zadanou cestou (proměnná @path) a vypíše tuto skutečnost do logovacího souboru

11.2 Zaplánování skriptu

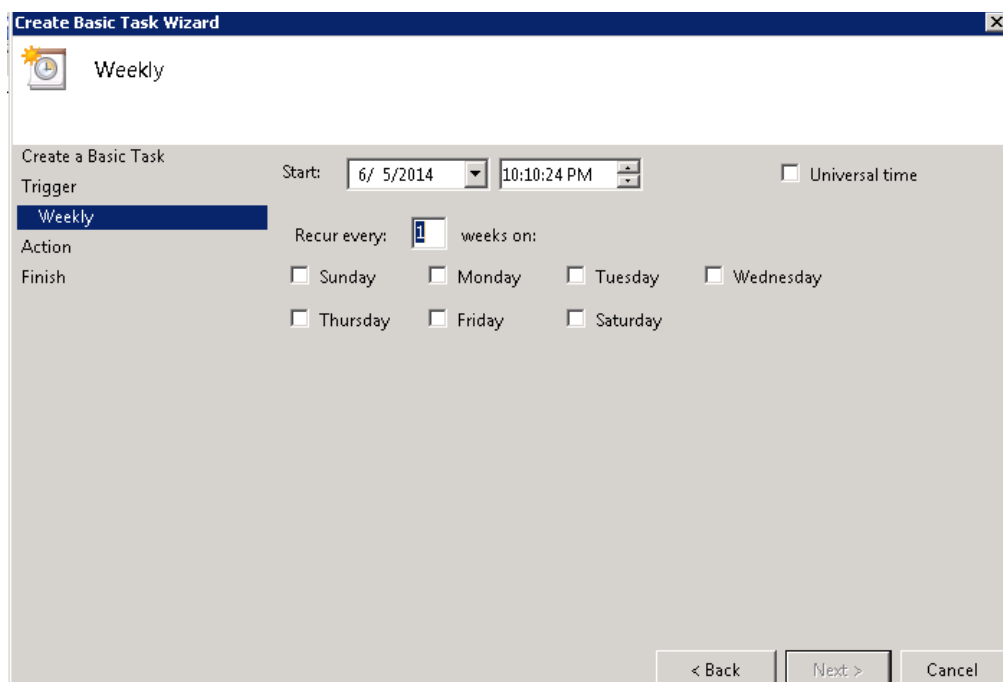
Nejprve byl spuštěn plánovač úloh pomocí Start -> Run -> Taskschd.msc -> create basic task. Dále bylo nastaveno spouštění jednou týdně s opakování vždy v pondělí. Byl vybrán skript pro spuštění. Na konci průvodce je jednoduché shrnutí a úloha je vytvořena. Postup je vidět na následující sekvenci obrázků.



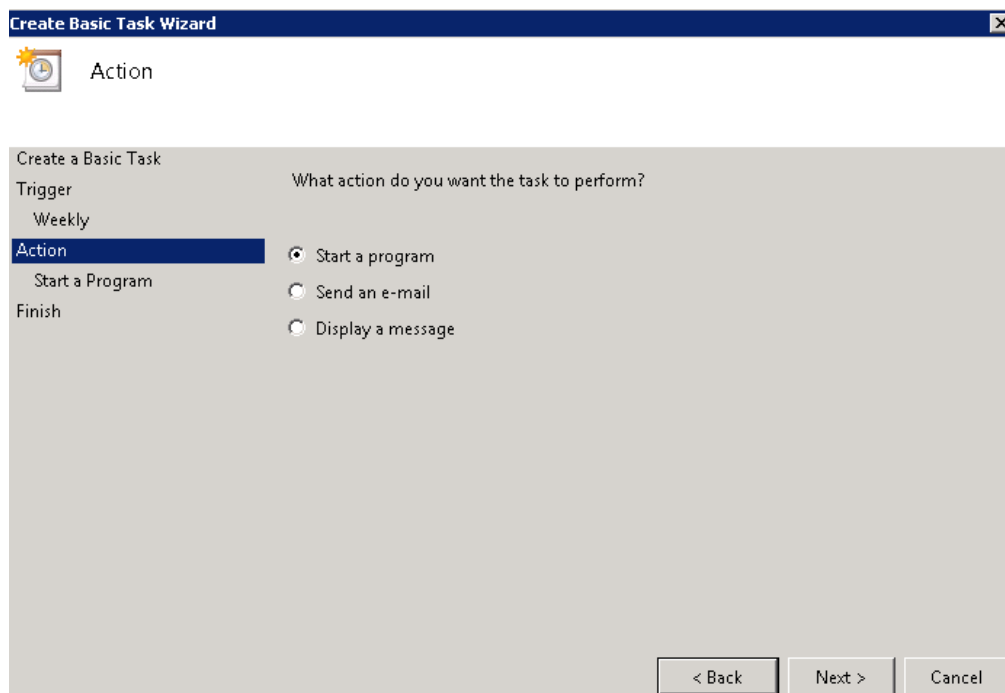
Obrázek 30: Vytvoření nové úlohy v plánovači úloh



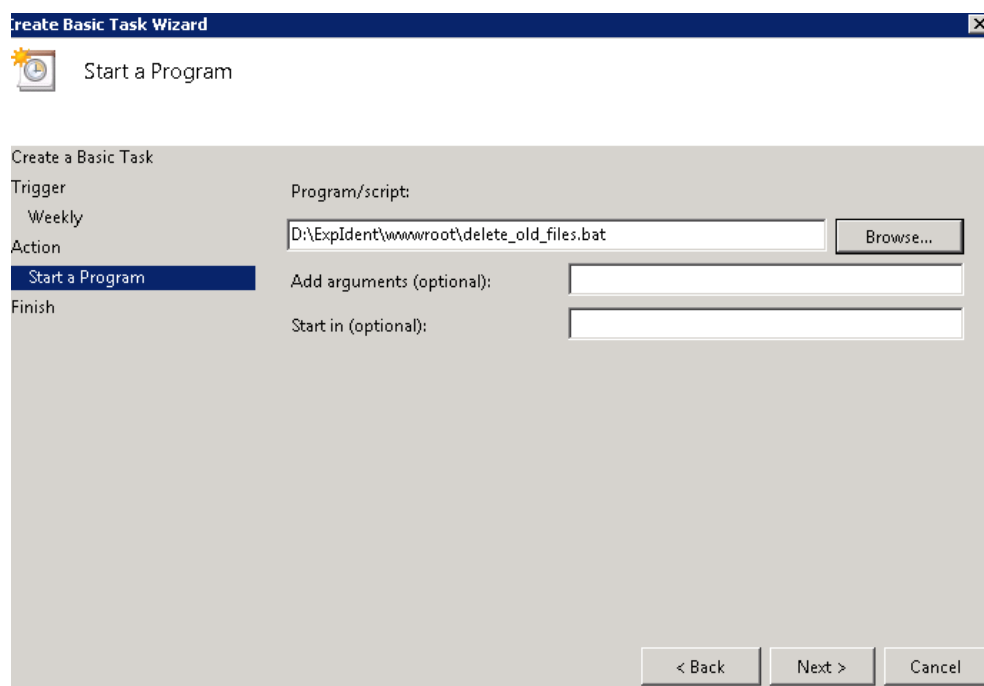
Obrázek 31: Zaplánování úlohy 1x týdně



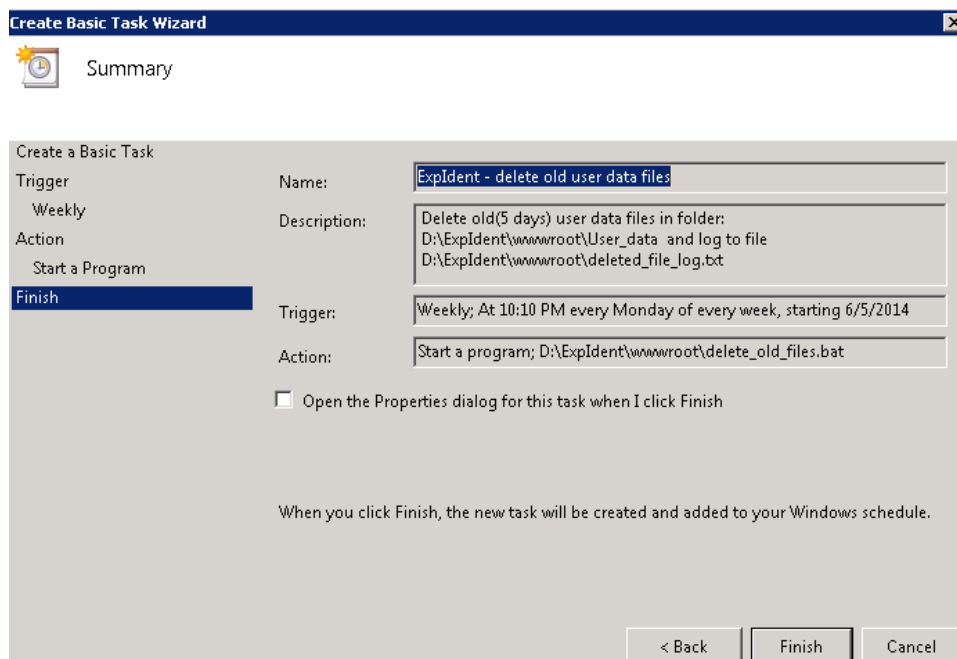
Obrázek 32: Úloha se bude provádět vždy v pondělí



Obrázek 33: Spouštět se bude program/skript



Obrázek 34: Výběr skriptu, který se bude spouštět



Obrázek 35: Sumář naplánované automatické úlohy

ZÁVĚR

Hlavním cílem této diplomové práce bylo vytvořit vícejazyčnou internetovou aplikaci pro identifikaci z experimentálních dat.

Úvodem diplomové práce je stručně popsána „příbuzná“ internetová aplikace pro návrh a ladění regulátorů z experimentálních dat, porovnány základní vlastnosti zmíněné a řešené aplikace.

V úvodní kapitole teoretické části jsou obecně popsány internetové aplikace, struktura internetových aplikací, webový server a vývojové prostředí webových aplikací. Stručně byl zmíněn účel a podstata takovýchto aplikací.

Další kapitola se již podrobněji věnuje technologiím použitých v jednotlivých vrstvách. Byly zmíněny pouze základy a nejčastěji využívané vlastnosti technologií, které jsou použity v této řešené aplikaci.

Navazující kapitoly popisují programové prostředí MATLAB a jsou zmíněny některé nekomerční alternativy. V následující kapitole je již popsán postup a nezbytné kroky pro propojení .NET komponent s internetovou aplikací.

Samostatná kapitola je věnována identifikaci, kde je popsána metoda nejmenších čtverců. Touto metodou je řešena identifikace z experimentálních dat v této řešené aplikaci. Vlastní algoritmus je naprogramovaný v programovém prostředí MATLAB.

V poslední kapitole teoretické části je stručně popsáno plánování úloh v systému Windows.

V praktické části je popsáno prostředí, které aplikace využívá pro svůj běh. o něco detailněji je popsáno uživatelské rozhraní řešené internetové aplikace. V podstatě se jedná o uživatelskou dokumentaci aplikaci. Každá část aplikace je popsána a vysvětlena k čemu slouží.

Nejrozsáhlejší blokem je vývojářská dokumentace. V této kapitole je popsáno kódové pozadí vytvořené aplikace. Vzhledem k velmi rozsáhlým zdrojovým kódům byly popsány pouze základní prvky. Podrobnější popis jednotlivých metod je uveden v přímo v komentářích zdrojového kódu. Detailněji je popsán postup, jakým byly m-soubory z programového prostředí MATLAB naimportovány do vývojového prostředí internetové aplikace napsané v ASP.NET a C#.

V předposlední kapitole jsou porovnány výsledky z řešené aplikace s výsledky získanými pomocí příkazu z *Identification toolbox* programového prostředí MATLAB.

Poslední kapitola popisuje nastavení automatického mazání starých uživatelských dat na serveru pomocí plánovače úloh Windows.

Internetová řešená aplikace splnila všechny požadované vlastnosti, které byly na začátku vývoje stanoveny. V průběhu vývoje aplikace se objevila celá řada problémů, které se z velké části podařilo vyřešit. Velmi častým jevem je údržba vytvořených internetových aplikací. Ani tato aplikace není výjimkou a vývoj není ukončen. Požadavky na úpravy a vylepšení, které nejsou součástí zadání této práce, můžou být řešeny v rámci jiné diplomové práce. Při větším rozšiřování aplikace by bylo vhodné implementovat některý z verzovacích systémů například velmi rozšířený a populární Git.

SEZNAM POUŽITÉ LITERATURY

Monografie:

- [1] BOBÁL, Vladimír. Identifikace systémů. Zlín: Univerzita Tomáše Bati ve Zlíně, 2009. ISBN 978-80-7318-888-7.
- [2] RAKUS, David. Web-aplikace pro přímý návrh a ladění regulátorů z experimentálních dat. Zlín, 2009. Diplomová práce. Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky.
- [3] PÍSEK, Slavoj. ASP.NET (začínáme programovat). Praha: Grada Publishing, 2003. ISBN 80-247-0526-5.
- [4] MICROSOFT. Vytváříme zabezpečené aplikace v Microsoft ASP.NET. Praha: Computer Press, 2004. ISBN 80-251-0466-4.
- [5] KARBAN, Pavel. Výpočty a simulace v programech Matlab a Simulink. Praha: BEN-technická literatura, 2007. ISBN 978-80-251-1448-3.
- [6] ZEMEK, Lukáš. Bezpečnost webových aplikací. Praha, 2012. 68 s. Bakalářská práce. Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky.
- [7] PROKOP, R.; MATUŠŮ, R.; PROKOPOVÁ, Z. *Teorie automatického řízení : lineární spojité dynamické systémy*. 1. Vydání. Zlín : UTB FAI ve Zlíně, 2006. 102 s. ISBN 80-7318-369-2..
- [8] PAYNE, Chris. *Naučte se ASP.NET za 21 dní*. Vyd. 1. Praha: Computer Press, 2002, xxii, 763 s. ISBN 80-722-6605-5.
- [9] MACDONALD, Matthew. *ASP.NET 2.0 a C#: tvorba dynamických stránek profesionálně*. Vyd. 1. Brno: Zoner Press, 2006, 1376 s. ISBN 80-868-1538-2.

Internetové zdroje:

- [10] GAZDOŠ, František a David RAKUS. MAT Server: Přímý návrh a ladění regulátorů z experimentálních dat [online]. c2009-2010 [cit. 2014-01-08]. Dostupné z: <http://matserver.utb.cz>.
- [11] Souborový systém. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-06-05]. Dostupné z: http://cs.wikipedia.org/wiki/Souborov%C3%BD_syst%C3%A9m

- [12] Databáze. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-06-05]. Dostupné z: <http://cs.wikipedia.org/wiki/Datab%C3%A1ze>
- [13] *Jak psát web* [online]. 2014 [cit. 2014-06-05]. Dostupné z: <http://www.jakpsatweb.cz/>
- [14] *Jak na plánovač úloh ve Windows 7* [online]. 2011 [cit. 2014-06-05]. Dostupné z: <http://www.winseven.cz/tipy-triky/jak-na-plnova-loh-ve-windows-7/>
- [15] The C# Programming Yellow Book. MILES, Rob. *Robmiles.com* [online]. [cit. 2014-06-09]. Dostupné z: <http://www.robmiles.com/s/Rob-Miles-CSharp-Yellow-Book-2014-m2e4.pdf>
- [16] 14 lessons to get you started with C# and .NET. RASHEED, Faraz. *Programmers heaven* [online]. [cit. 2006-12-05]. Dostupné z: http://ebooks.programmersheaven.com/csharp_ebook.pdf

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

HTTP	Hypertext Transfer Protocol
AJAX	Asynchronous JavaScript and XML
ASP	Active Server Pages
CSS	Cascading Style Sheets
IIS	Internet Information Services
HTML	HyperText Markup Language
XHTML	eXtensible Hypertext Markup Language
CLR	Common language Runtime
MATLAB	MAtrix LABoratory
GPL	General Public License

SEZNAM OBRÁZKŮ

Obrázek 1: struktura internetové aplikace	12
Obrázek 2: .NET <i>framework</i>	15
Obrázek 3: Zdrojový M-soubor pro funkci na vykreslení grafu ze zadaných dat	20
Obrázek 4: Deployment Tool	21
Obrázek 5: Složka Bin a reference na MATLAB knihovnu.....	22
Obrázek 6: vytvoření instance třídy z importované knihovny.....	22
Obrázek 7: Blokové schéma modelu obecného stochastického procesu	23
Obrázek 8: Obecné tvary polynomů	24
Obrázek 9: model ARX	24
Obrázek 10: Blokové schéma ARX modelu.....	25
Obrázek 11: Plánovač úloh.....	29
Obrázek 12: Základní informace o serveru.....	31
Obrázek 13: Ukázka Visual Studio Web Developer 2008 Express Edition	32
Obrázek 14: Úvodní stránka aplikace.....	33
Obrázek 15: Výběr a import souboru	34
Obrázek 16: Výběr a načtení jednotlivých proměnných ze souboru	35
Obrázek 17: Filtrace uživatelských dat.....	35
Obrázek 18: Karta pro zadání stupňů polynomů ARX modelu.....	36
Obrázek 19: Model identifikované soustavy – testovací data	37
Obrázek 20: Karta pro zobrazení spojitého modelu	38
Obrázek 21: Spojitý model a porovnání průběhu identifikovaného modelu a reálných dat.....	39
Obrázek 22: Přepínač jazyka	39
Obrázek 23: Struktura aplikace.....	41
Obrázek 24: Deployment tool.....	46
Obrázek 25: Úvodní stránka internetové aplikace.....	47
Obrázek 26: Seznam pro výběr jazyka v souboru LangMenu.ascx.....	49
Obrázek 27: Ukázka struktury lokální zdrojových souborů pro multijazyčnost	50
Obrázek 28: Simulační schéma z prostředí Simulink.....	52
Obrázek 29: Chování identifikovaného modelu z řešené aplikace v porovnání s výstupem funkce arx.....	53
Obrázek 30: Vytvoření nové úlohy v plánovači úloh.....	55

Obrázek 31: Zaplánování úlohy 1x týdně.....	56
Obrázek 32: Úloha se bude provádět vždy v pondělí	56
Obrázek 33: Spouštět se bude program/skript	57
Obrázek 34: Výběr skriptu, který se bude spouštět	57
Obrázek 35: Sumář naplánované automatické úlohy	58

SEZNAM PŘÍLOH

P I CD-ROM

PŘÍLOHA P I: CD-ROM

Na přiloženém CD-ROM jsou umístěny všechny potřebné zdrojové soubory, které jsou nutné pro nahrání aplikace na server. Dále je na přiloženém médiu vlastní text Diplomové práce.

Vzhledem k rozsáhlosti zdrojových kódů nebudou uvedeny v přílohách. Dostupné budou pouze online nebo na přiloženém médiu.