

# **Návrh aplikace pro podporu výuky Matlabu a Simulinku pro kombinované studium**

Patrik Kadlečík



## **ZADÁNÍ BAKALÁŘSKÉ PRÁCE**

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Patrik Kadlečík**  
Osobní číslo: **A11118**  
Studijní program: **B3902 Inženýrská informatika**  
Studijní obor: **Informační a řídicí technologie**  
Forma studia: **prezenční**

Téma práce: **Návrh aplikace pro podporu výuky Matlabu a Simulinku pro kombinované studium**

Téma anglicky: **The Design of an Application in Support of the Tuition of the MATLAB and Simulink Subjects for the Combined Studies/Distance Curriculum**

Zásady pro vypracování:

1. Vypracujte literární rešerži na dané téma.
2. Seznamte se s prací v programu Matlab a Simulink.
3. Analyzujte již existující podpory výuky pro program Matlab a Simulink.
4. Vytvořte vlastní návrh aplikace pro podporu výuky v programu Matlab a Simulink pro kombinované studium.
5. Vytvořte příklady v programu Matlab a Simulink a umístěte je na DVD.

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. PERŮTKA, Karel. MATLAB: základy pro studenty automatizace a informačních technologií. Vyd. 1. Zlín: Ústav řízení procesů, Institut řízení procesů a aplikované informatiky, Fakulta technologická, Univerzita Tomáše Bati ve Zlíně, 2005, 303 s. ISBN 80-7318-355-2.
2. DUŠEK, František. Matlab a Simulink: úvod do používání. 1.vyd. Pardubice: Univerzita Pardubice, 2001, 146 s. ISBN 80-7194-273-1.
3. ZAPLATÍLEK, Karel a Bohuslav DOŇAR. MATLAB pro začátečníky. 2. vyd. Praha: BEN – technická literatura, 2005, 151 s. ISBN 80-7300-175-6.
4. ZAPLATÍLEK, Karel a Bohuslav DOŇAR. MATLAB: tvorba uživatelských aplikací. 1. vyd. Praha: BEN – technická literatura, 2004, 215 s. ISBN 80-7300-133-0.
5. KOZÁK, Štefan a Slavomír KAJAN. MATLAB-SIMULINK 1. 1. vyd. Bratislava: Slovenská Technická Univerzita v Bratislavě, 1999, 125 s. ISBN 80-277-1213-2.

Vedoucí bakalářské práce: **Ing. Karel Perůtka, Ph.D.**  
Ústav řízení procesů

Datum zadání bakalářské práce: **6. března 2015**

Termín odevzdání bakalářské práce: **22. května 2015**

Ve Zlíně dne 6. března 2015



doc. Mgr. Milan Adámek, Ph.D.  
*děkan*



L.S.



prof. Ing. Vladimír Vašek, CSc.  
*ředitel 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 připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- 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/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....  
podpis diplomanta

## **ABSTRAKT**

Táto bakalárska práca sa zaoberá tvorbou výučbového materiálu pre prácu v programe MATLAB a Simulink.

Hlavným cieľom je vytvoriť prehľad základných príkladov (príkazov) a navrhnúť výučbovú aplikáciu vhodnú pre študentov, ktorí začínajú s programom MATLAB a Simulink.

V teoretickej časti je vykonaná analýza literárnych a výučbových zdrojov z prostredia MATLAB a Simulink. Praktická časť je zameraná na popis základných príkazov v programe MATLAB, ktoré slúžia pre vysvetlenie funkčnosti a syntaxie jednotlivých príkazov. Na konci každej kapitoly sú vytvorené príklady k precvičeniu danej problematiky. V závere bakalárskej práce je popísaná tvorba grafického programu v prostredí MATLAB (GUI), ktorý slúži pre podporu výučby, najmä pre študentov kombinovanej formy štúdia.

Kľúčové slova: MATLAB, Simulink, príkaz, vetvenie, syntaxia

## **ABSTRACT**

This Bachelor's Thesis deals with the creation of educational materials for working with the MATLAB and Simulink software.

The main objective is to create a summary of basic examples (commands) and to suggest educational application suitable for students who are beginning to work with the MATLAB and Simulink software.

In the theoretical part, the analysis of literary and educational resources from MATLAB and Simulink user interface is performed.

The practical part is focused on description of basic commands in MATLAB software, which are used for explaining the functionality and syntax of individual commands. At the end of each chapter, exercises for practising the issue are designed.

The conclusion of the Bachelor's Thesis describes the creation of graphics software in MATLAB (GUI), which serves as learning support, especially for students of the combined form of study.

Keywords: MATLAB, Simulink, Command, Branching, Syntax

**„Když všichni mluví o nemožnostech, hledej možnosti.“**

**Tomáš Baťa**

### **PodĎakovanie**

Rád by som touto cestou poďakoval Ing. Karolovi Perůtkovi Ph.D. za odborné vedenie, čas a pripomienky pri spracovávaní bakalárskej práce.

Ďalej by som rád poďakoval svojej priateľke Lucii, ktorá bola mojou najväčšou podporou pri štúdiu a písaní tejto bakalárskej práce. Moje poďakovanie patrí aj rodičom za podporu po celú dobu štúdia.

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



## OBSAH

<b>ÚVOD.....</b>	<b>10</b>
<b>I TEORETICKÁ ČASŤ.....</b>	<b>11</b>
<b>1 O PROGRAME MATLAB A SIMULINK.....</b>	<b>12</b>
1.1 MATLAB.....	12
1.2 SIMULINK.....	14
1.3 VYUŽITIE MATLABU A SIMULINKU .....	15
1.4 VZNIK A HISTÓRIA MATLABU.....	16
<b>2 POPIS PROGRAMOVACIEHO PROSTREDIA MATLAB.....</b>	<b>18</b>
2.1 POPIS PRACOVNEJ PLOCHY .....	18
2.1.1 Menu (Položka 1).....	18
2.1.2 Aktuálna zložka (Položka 2).....	18
2.1.3 Current Folder (Položka 3).....	19
2.1.4 Command Window (Položka 4).....	19
2.1.5 Workspace (Položka 5) .....	19
2.1.6 Command History .....	19
2.1.7 Nápoveda.....	20
2.2 VYTVORENIE SKRIPTU (M-FILES) v MATLABE .....	21
2.3 PRÁCA V PRÍKAZOVOM RIADKU.....	22
2.4 TYPY MATLAB SÚBOROV REGISTROVANÉ V MS WINDOWS .....	23
2.4.1 Súbory typu .FIG.....	23
2.4.2 Súbory typu .M.....	23
2.4.3 Súbory typu MAT .....	23
2.4.4 Súbory typu .P.....	23
2.4.5 Súbory typu MEX .....	24
<b>3 POPIS PROGRAMOVACIEHO PROSTREDIA SIMULINK.....</b>	<b>25</b>
3.1 SPUSTENIE SIMULINKU.....	25
3.2 SUBSYSTÉMY A MASKA SUBSYSTÉMU .....	26
<b>4 PREMENNÉ .....</b>	<b>28</b>
4.1 LOKÁLNE PREMENNÉ.....	29
4.2 GLOBÁLNE PREMENNÉ .....	29
4.3 PRESISTENTNÉ PREMENNÉ.....	30
4.4 REZERVOVANÉ KLÚČOVÉ SLOVÁ.....	30
<b>5 DATOVÉ TYPY .....</b>	<b>31</b>



5.1	CELOČÍSELNÉ DÁTOVÉ TYPY .....	31
5.2	ČÍSELNÝ DÁTOVÝ TYP DOUBLE .....	31
5.3	ČÍSELNÝ DÁTOVÝ TYP SINGLE .....	31
5.4	DÁTOVÉ TYPY UINT8, UINT16, UINT32 A UINT64.....	32
5.5	DÁTOVÉ TYPY INT8, INT16, INT32, INT64 .....	32
5.6	DÁTOVÝ TYP CHAR.....	33
5.7	DÁTOVÝ TYP CELL.....	34
5.8	DÁTOVÝ TYP STRUCT .....	35
5.9	DÁTOVÝ TYP HANDLE FUNCTION @ .....	36
5.10	KOMPLEXNÉ ČÍSLA .....	37
<b>6</b>	<b>OPERÁTORY .....</b>	<b>38</b>
6.1	ARITMETICKÉ OPERÁTORY .....	38
6.2	RELAČNÉ OPERÁTORY .....	39
6.3	LOGICKÉ OPERÁTORY .....	39
6.4	ŠPECIÁLNE ZNAKY .....	40
6.4.1	Priorita operátorov .....	40
<b>7</b>	<b>PODMIENKY A CYKLY .....</b>	<b>41</b>
7.1	PODMIENKA IF.....	41
7.2	MNOHONÁSOBNÉ VETVENIE SWITCH.....	41
7.3	CYKLY .....	42
7.3.1	Cyklus while.....	42
7.3.2	Cyklus for.....	43
7.3.3	Príkazy continue, break a return.....	44
7.4	POLIA, VEKTORY A MATICE .....	44
7.5	SKALÁR.....	45
7.6	FUNKCIA .....	46
<b>II</b>	<b>PRAKTICKÁ ČASŤ .....</b>	<b>47</b>
<b>8</b>	<b>ZÁKLADNÉ OPERÁCIE S VEKTORMI A MATICAMI .....</b>	<b>48</b>
8.1	VEKTORY .....	48
8.2	MATICE .....	49
8.2.1	Výber prvkov a určitej časti matice .....	54
8.2.2	Funkcie pre manipuláciu s maticami.....	56
8.2.3	Maticové operácie .....	59
8.2.4	Riešenie sústav lineárnych algebraických rovníc.....	64
8.3	NERIEŠENÉ PRÍKLADY .....	66
<b>9</b>	<b>KOMPLEXNÉ ČÍSLA.....</b>	<b>67</b>
9.1	NERIEŠENÉ PRÍKLADY .....	69
<b>10</b>	<b>PROGRAMOVANIE PODMIENOK A CYKLOV .....</b>	<b>70</b>

10.1	PODMIENKA IF .....	70
10.2	VETVENIE POMOCOU SWITCH .....	70
10.3	CYKLUS WHILE.....	74
10.4	CYKLUS FOR.....	75
10.5	NERIEŠENÉ PRÍKLADY .....	76
<b>11</b>	<b>REŤAZCE.....</b>	<b>77</b>
11.1	NERIEŠENÉ PRÍKLADY .....	79
<b>12</b>	<b>GRAFY.....</b>	<b>80</b>
12.1	2D GRAFY .....	80
12.2	NAJPOUŽÍVANEJŠIE GRAFY V OBLASTI AUTOMATIZÁCIE A TEÓRIE SYSTÉMU .....	83
12.2.1	Impulzová charakteristika .....	83
12.2.2	Prechodová charakteristika .....	84
12.2.3	Nyquistova krivka .....	85
12.2.4	Bodeho krivka .....	85
12.3	NERIEŠENÉ PRÍKLADY .....	86
<b>13</b>	<b>FUNKCIE.....</b>	<b>87</b>
<b>14</b>	<b>DÁTOVÉ SÚBORY .....</b>	<b>90</b>
14.1	PRÁCA S ADRESÁRMÍ A SÚBORMI.....	90
14.1.1	Práca s adresármi.....	90
14.1.2	Práca so súbormi .....	92
14.2	IMPORT A EXPORT DO MAT-SÚBOROV .....	92
14.2.1	Export do MAT- súborov .....	93
14.2.2	Import z MAT- súborov .....	93
14.3	IMPORT TEXTOVÝCH SÚBOROV.....	94
14.4	EXPORT DO TEXTOVÝCH SÚBOROV.....	96
14.5	IMPORT A EXPORT XLS .....	96
14.5.1	Informácie o súbore.....	97
14.5.2	Export do XLS súboru .....	97
14.5.3	Import zo XLS súboru.....	98
<b>15</b>	<b>SIMULINK .....</b>	<b>100</b>
15.1	JEDNODUCHÉ PRÍKLADY V SIMULINKU .....	101
<b>16</b>	<b>GRAFICKÉ ROZHRAŇIE V MATLABE A SIMULINKU .....</b>	<b>103</b>
16.1	APLIKÁCIA PRE PODPORU VÝUČBY PROGRAMU MATLABU .....	103
16.1.1	Popis grafického rozhrania aplikácie .....	104
<b>ZÁVER .....</b>		<b>119</b>
<b>ZOZNAM POUŽITEJ LITERATURY .....</b>		<b>120</b>
<b>ZOZNAM OBRÁZKOV .....</b>		<b>122</b>
<b>ZOZNAM TABULIEK .....</b>		<b>124</b>
<b>ZOZNAM PRÍLOH .....</b>		<b>124</b>

## ÚVOD

Dnes existuje viacero programov, ktoré sú schopné vykonať realizáciu numerických výpočtov ako je Mathematica, Scilab alebo MATLAB. MATLAB patrí k najpoužívanejším programom, ktorý umožňuje počítanie s maticami, vykresľovanie 2D i 3D grafov, funkcií, implementáciu algoritmov, počítačových simulácií, analýzu a prezentáciu dát i vytváranie aplikácií vrátane užívateľského rozhrania. S programom MATLAB je spojený program Simulink, ktorý využíva funkcie MATLABu pre simuláciu dynamických systémov. Simulink je jednoduchší a intuitívnejší, ale nie je ho možné používať bez znalosti MATLABu.

Hlavným cieľom práce je vytvoriť prehľad základných príkladov (príkazov) a navrhnúť výučbovú aplikáciu vhodnú pre študentov, ktorí začínajú s programom MATLAB a Simulink.

Najprv je v práci popísaný MATLAB od jeho programovacieho prostredia, kde sú vysvetlené jednotlivé položky a súbory, s ktorými sa v prostredí MATLAB dá pracovať. V teoretickej časti je ďalej popísané programovacie prostredie Simulinku. Za touto kapitolou sú ďalej vysvetlené premenné, dátové typy, operátory, podmienky a cykly, s ktorými sa čitateľ v rámci programovania v MATLABe stretne.

Praktická časť nadväzuje na časť teoretickú a obsahuje kapitoly, v ktorých študent nájde riešené príklady v prostredí MATLAB i Simulink. Práca obsahuje neriešené príklady určené k zopakovaniu daného okruhu. V praktickej časti sa nachádzajú riešené príklady práce s maticami, vektormi, komplexnými číslami, podmienkami, cyklami, reťazcami, grafmi, funkciami a dátovými súbormi.

V závere bakalárskej práce je popísaná tvorba aplikácie pre podporu výučby v MATLABe a Simulinku pre kombinovaných študentov, ktorý bola vytvorený v grafickej časti programu MATLAB a slúži študentom k interaktívnej výučbe v rámci predmetu MATLAB a Simulink.

## **I. TEORETICKÁ ČÁST**

## 1 O PROGRAME MATLAB A SIMULINK

V tejto kapitole bude predstavený programovací jazyk MATLAB a jeho nadstavba Simulink.

### 1.1 Matlab

Názov MATLAB podľa Duška (2000, s. 9) a Karbana (2006, s. 11) vznikol z anglických slov MATrix LABoratory. Programovací jazyk MATLAB sa vyvinul z Linpack a EISPACK matice algoritmov. (Wirth a Kovesi, 2005, s. 1)

Definícia MATLABu podľa firemnej literatúry je nasledujúca:

*„MATLAB je vysoko výkonný jazyk pre technické výpočty. Integruje výpočty, vizualizáciu a programovanie do jednoducho použiteľného prostredia, kde problémy i riešenie, sú vyjadrené v prirodzenom tvare“* (Dušek, 2000, s. 9)

Dušek (2000, s. 9) a Karban (2006, s. 11) sa zhodujú, že MATLAB je výkonné interaktívne prostredie pre vedecké výpočty, ktorého základným dátovým typom je dvojrozmerné pole, spájajúce technické výpočty, modelovanie, vizualizáciu dát, grafiku, návrhy algoritmov a programovací jazyk v jednom prostredí.

MATLAB obsahuje jednoduché i zložité matematické funkcie, ktoré sú obsiahnuté v algoritmoch. Tieto funkcie, ktoré sú schopné vyriešiť rôzne okruhy problémov, sa nazývajú Toolboxy, čo sú problémovo orientované súbory funkcií pre riešenie problematiky určitých oborov. (Dušek, 2000, s. 9) Toolboxy ako nástroje, obsahujú kód, aby mohli byť riešené špecifické oblasti použitia, popísal Perůtka (2005, s. 12)

Toolboxy, ako nadstavby programov uvádza Perůtka (2005, s. 13) napr.:

Database Toolbox - súbory pre prenos dát medzi MATLABom a databázovými formátmi ako Microsoft Access, Oracle, Microsoft SQL alebo IBM DB2.

Financial ToolBox - nástroj pre ekonomické a finančné aktivity.

Excel Link for MATLAB - dodatkový software umožňujúci integráciu MATLAB a MS Excel.

Mapping toolbox for MATLAB - umožňuje čítanie, analýzu a zobrazenie geografických dát v MATLABe.

Programovací jazyk je vhodný pre inžinierov, vedcov, matematikov i učiteľov. (Karban, 2006, s. 12) Perůtka (2005, s. 8) a Karban (2006, s. 12) uvádzajú, že v posledných rokoch sa tento programovací jazyk stal vhodnou pomôckou pri výučbe študentov na vysokých školách, a to predovšetkým preto, že obsahuje množstvo dostupných modulov. Wirth a Kovesi (2005, s. 1) dopĺňujú, že program MATLAB dokáže ušetriť študentovi až o 50 % viac času, pokiaľ ho využíva k riešeniu najrôznejších problémov napr. k rozvoju algoritmov a analýz, modelovanie, simulácie, prototypovanie, vedecké a inžinierske výpočty a vizualizácie.

MATLAB je jazyk skriptovací, ktorý predstavuje veľmi odlišný štýl programovania než u systémových programovacích jazykov, ako je C alebo Java. Skriptovací jazyky používajú prístupy, ktoré vedú k dosiahnutiu vyššej úrovne programovania a rýchlejšiemu vývoju aplikácii než systémovo programovateľné jazyky. (Wirth a Kovesi, 2005, s. 2-3)

Karban (2006, s. 12) dopĺňuje, že sa jedná o programovací jazyk, ktorý ponúka výhodu v tom, že je jednoduchší než jeho konkurenti, pretože obsahuje vstavané funkcie pre ľahší vývoj. MATLAB podporuje komplexné čísla, prácu s maticami, rieši lineárne i diferenciálne rovnice.

Najpodstatnejšiu časť numerického jadra MATLABU tvoria algoritmy pre operácie s maticami reálnych a komplexných čísel. MATLAB je možné využiť pre najjednoduchšie operácie až po zložité dátové typy a dátové štruktúry. V MATLABe môže vektor reálnych čísel predstavovať aj polynóm, ktorý program taktiež obsahuje. Ďalej je možné v programe pracovať taktiež s objektmi, ktoré užívateľovi umožňujú rozšíriť výpočtové prostredie o nové dátové typy, na ktorých je možno definovať ľubovoľné funkcie alebo operátory. (Perůtka, 2005, s. 8)

Stavba MATLABu je syntaktická, ako u väčšiny z mnohých programovacích jazykov napr. Fortran, Java alebo C. Avšak vo zrovnaní s inými programovacími jazykmi ako je Java alebo C, je výhoda programacieho jazyka MATLAB zakotvená v jeho univerzálnosti a efektívnom rozvoji v prieskumoch alebo prototypovanie v jednotlivých etapách riešenia problémov. (Wirth a Kovesi, 2005, s. 2-3)

MATLAB používa zjednodušenú syntaktickú štruktúru, ktorá:

- napomáha k pochopeniu, ako software interaguje so systémami,
- poskytuje komplexnú testovaciu kontrolu,

- môže byť použitá k ilustrácii rolí programovaní pri riešení vedeckých a technických problémov. (Wirth a Kovesi, 2005, s. 2-3)

Medzi kľúčové vlastnosti Karban (2006, s. 12) zaradil:

- vysokoúrovňový jazyk pre technické výpočty,
- otvorený a rozšíriteľný systém,
- veľké množstvo aplikačných knižníc,
- grafické užívateľské rozhranie,
- podpora viacrozmerných polí,
- import a export dát do rôznych formátov,
- rozšíriteľnosť modulov jazykmi C, C++, Fortran a Java.

Dostupné rozšírenia programu MATLAB zachytáva Obr. 1.



Obr. 1 Dostupné rozšírenia Matlabu (Karban, 2006, s. 12)

## 1.2 Simulink

Názov Simulink vznikol z anglických slov SIMUlation and LINK čo c slovenskom preklade znamená Simulácia a spojenie. Karban (2006, s. 11) definoval Simulink ako grafické prostredie pre simuláciu a najrozšírenejší toolbox určený predovšetkým pre simuláciu dynamických systémov. Podľa Kupky (2007, s. 116) je Simulink najznámejšia a najpoužívanjšia nadstavba MATLABu pre simuláciu a modelovanie dynamických systémov v prehľadnom grafickom užívateľskom prostredí (GUI), ktorý využíva algoritmy MATLABu pre numerické riešenie nelineárnych diferenciálnych rovníc. Vytvára pre užívateľa možnosť rýchlo a ľahko vytvárať modely dynamických sústav vo forme blokových schém, ktoré sú podobné zapojeniu pre analógový počítač, a rovníc. (Humusoft, ©1991-2015)

Karban (2006, s. 163) dopľňuje, že modely užívateľ definuje jednoduchým spôsobom prostredníctvom jednotlivých blokov z knižnice pomocou tzv click-and-drag operácie myšou,

teda kliknutím na blok a pretiahnutím na požadované miesto. Simulink obsahuje rozsiahlu knižnicu blokov. Simulink poskytuje aj možnosť vytvoriť si vlastné bloky.

Simulink je určený na časové riešenie – simulácie – chovanie dynamického systému pokiaľ poznáme matematický popis daného systému. Je teda možné určiť časové priebehy výstupných veličín (i všetkých vnútorných) v závislosti na časových priebehoch vstupných a počiatočných stavov. Simulink sa taktiež využíva pri elektrických, mechanických či termodynamických systémov. (Dušek, 2000, s. 108)

Simulink využíva rozsiahle grafické možnosti Windowsu. Algoritmy MATLABu sú pre Simulink využívané pre numerické riešenie diferenciálnych rovníc. (MathWorks, ©1994-2015)

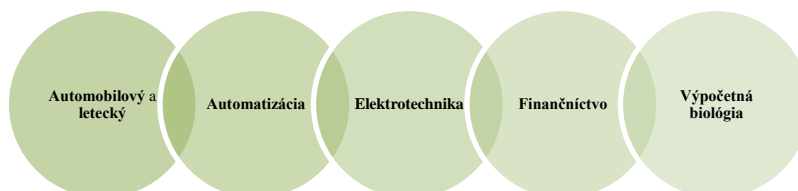
### 1.3 Využitie MATLABu a Simulinku

Využitie MATLABu a Simulinku má široké spektrum v mnohých oblastiach. Umožňuje konštrukciu a vývoj širokého spektra moderných produktov, vrátane automobilových systémov, leteckých riadení letu, telekomunikácií a ďalších elektronických zariadení, priemyslových strojov a zdravotníckych prostriedkov. (MathWorks, ©1994-2015)

V prostredí MATLAB by užívateľ mal vedieť základné 3 pravidlá:

- čo chcem - jasne formulovať problém,
- ako toho dosiahnuť - mať predstavu o tom, ako sa problém rieši,
- ako overiť výsledok - vedieť odhadnúť, či je výsledok správny.

Priemyselné odvetvia s najčastejším využitím programu MATLAB a Simulink zobrazuje Obr. 2.



Obr. 2 Odvetvia priemyslu s najčastejším využitím programu MATLAB a Simulink (Karban, 2006, s. 11)



Ďalšie oblasti využitia sú aplikovaná matematika, automatické riadenie a regulácia, spracovanie signálu a komunikácie, spracovanie obrazu, meranie a testovanie, výpočtová biológia, finančné modelovanie a analýza. (Humusoft, ©1991-2015)

## 1.4 Vznik a história MATLABu

V roku 1970 bol MATLAB vytvorený Clevom Morelom na univerzite v Novom Mexiku. V nasledujúcich rokoch sa Cleve Morel spojil s Stevenom Bangertom a Jackom Littlom a prekódovali MATLAB do jazyka C a nazvali ho JACKPAC. V roku 1984 predali aplikáciu MATLAB americkej spoločnosti MathWorks, ktorá sa postarala o jeho ďalší vývoj. Pôvodne bol MATLAB zostavený pod OS Unix predovšetkým pre Unix.

V roku 1985 vznikla prvá verzia pre PC XT, nasledovala verzia MATLAB 386 s vnútornou virtuálnou pamäťou.

V roku 1994 vznikla verzia MATLABu pre Windows s kvalitnejším grafickým rozhraním a v roku 1995 MATLAB 386g definitívne ukončil vývoj pre OS MS DOS.

V roku 1999 bola verzia plne 32bitová.

V roku 2002 prišla na trh verzia s prepracovanou grafikou, kde sa súčasťou programu stal i Notebook, ktorý umožňoval spoluprácu s programom MS Word, čo umožnilo ďalšie prepojenie s jazykom Java. (Dušek, 2000, s. 11-12)

Verzie MATLAB a Simulink sa neustále vyvíjajú v posledných 10 rokoch v pravidelných dvoch verziách v jednom roku rozdelených na označenia písmenami A a B. (Kozák a Kajan, 1999, s. 13)

Najnovšou verziou je MATLAB R2014b, ktorý obsahuje:

- nový grafický systém,
- dátové typy pre reprezentáciu dát a času,
- možnosť zabaliť vlastný toolbox do jediného,
- funkciu pre spracovanie rozsiahlych dát s možnosťou využitia platformy Hadoop,
- integráciu so systémami Git a Subversion, prístup k projektom GitHub cez File Exchange a podporu prepojenia s platformami Arduino a Android. (Humusoft, ©1991-2015)

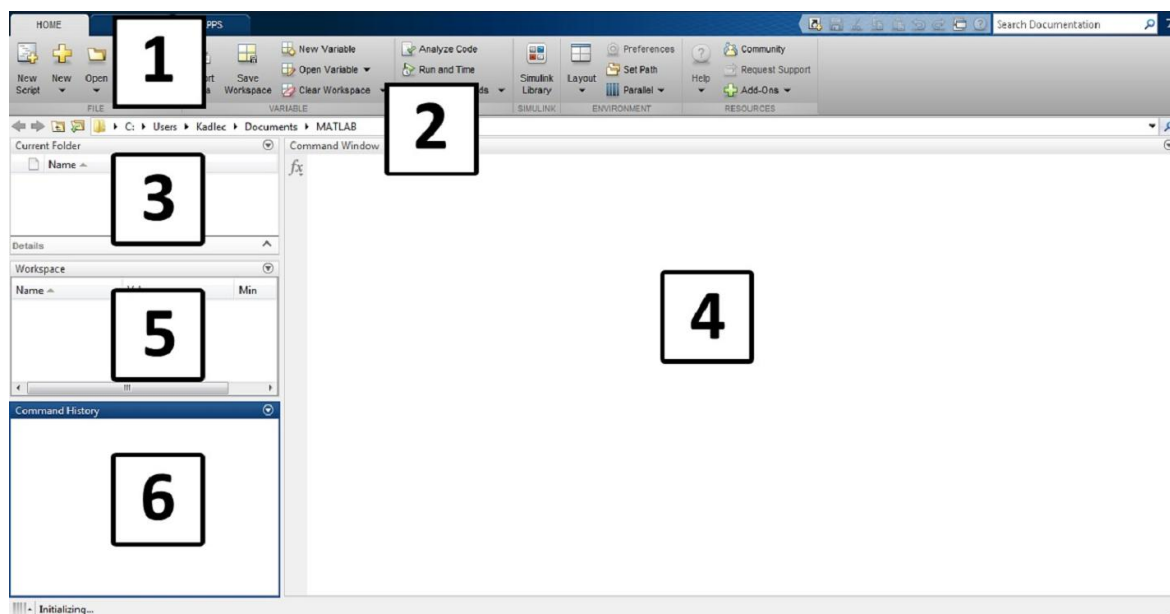
Medzi novinky v Simulinku patrí:

- vodiace značky a nápovedy pre rýchlejšiu tvorbu modelov,
- grafické spojenie textových poznámok s blokmi a špeciálny náhľad v editore pre zobrazenie rozhrania modelov a subsystémov,
- tlačidlo pre rýchly reštart po sebe idúcich simuláciách,
- živé zobrazenie signálových dát a ďalšie nové funkcie v Simulation Data Inspector,
- nový blok Simulink Function pre tvorbu a volanie modelovaných funkcií odkiaľkoľvek zo Simulinku a Stateflow. (Humusoft, ©1991-2015)

Dnes má spoločnosť MathWorks sídlo v Massachusetts v USA a pôsobí v 15 krajinách sveta. (MathWorks, ©1994-2015)

## 2 POPIS PROGRAMOVACIEHO PROSTREDIA MATLAB

Po spustení programu MATLAB sa zobrazí pracovná plocha viz Obr. 3.



Obr. 3 Pracovná plocha MATLAB (Vlastné spracovanie)

### 2.1 Popis pracovnej plochy

Popis jednotlivých častí pracovnej plochy, ktorá sa zobrazí po spustení programu MATLAB.

#### 2.1.1 Menu (Položka 1)

Ponuka menu je rozdelená do základných troch kategórií. V záložke HOME sa nachádzajú základné ponuky ako vytvorenie nového súboru, načítanie súborov, pridanie a odobranie prídavných okien. Čiastočne je to náhrada aj za tlačidlo ŠTART, ktoré sa nachádzalo v ľavom dolnom rohu. Vývojári premiestnili všetky položky z panela ŠTART do menu. V záložke PLOTS sa nachádzajú rôzne druhy grafov, ktoré si užívateľ v prípade potreby môže vybrať. Treťou záložkou je APPS, kde sa nachádzajú aplikácie, ktoré uľahčujú užívateľovi prácu s PID regulátorom a analýzou signálu. Podľa Kupky (2007, s.11) sa do nových verzií MATLABu dajú jednoducho tieto aplikácie doinštalovať z internetu.

#### 2.1.2 Aktuálna zložka (Položka 2)

Slúži pre nastavenie cesty k aktuálnej zložke.

### 2.1.3 Current Folder (Položka 3)

Hlavný nástroj pre prácu s MATLAB súbormi. Umožňuje prístup operačnému systému do správy súborov v rámci MATLABu. (Perůtka, 2005, s. 17)

### 2.1.4 Command Window (Položka 4)

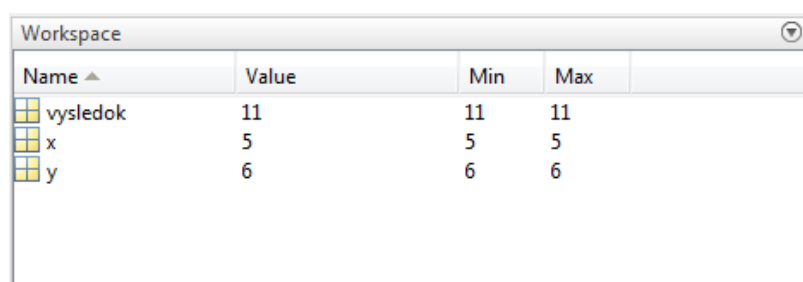
Je jeden z najhlavnejších a najdôležitejších častí pracovnej plochy (desktopu). V tomto okne zadáva užívateľ jednotlivé príkazy, spúšťa MATLAB funkcie a M-FILE súbory. V tomto okne je zobrazený výsledok a rôzne systémové hlásenie MATLABU. (Kupka, 2007, s. 11)

MATLAB bol pôvodne MS DOSový príkazový interpretér, preto neobsahuje žiadne menu alebo sady ikon pre zadávanie príkazov ako napr. MS Word alebo AutoCAD. Všetko sa píše ručne. (Perůtka, 2005, s. 17)

### 2.1.5 Workspace (Položka 5)

V preklade „pracovný priestor“ je súbor premenných, ktoré už boli použité pri práci v MATLABe. Workspace je pri prvom otvorení MATLABu prázdne. Poklepaním myšou na niektorú z premenných je možné zobrazit' detailné informácie. (rozmer, štruktúra, atd). (Kupka, 2007, s. 11)

Do vytvorenej premennej x a y sa zapisujú hodnoty 5 a 6. Do premennej výsledok sa uloží súčet premenných x a y. Všetky premenné sú uložené do Workspace viz Obr. 4.



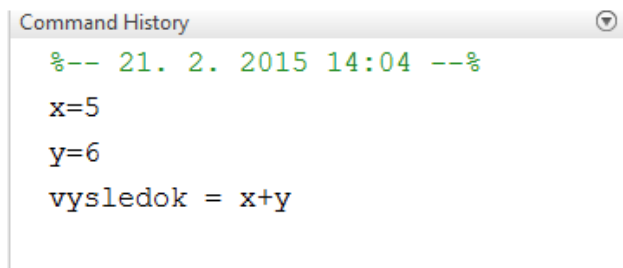
Name	Value	Min	Max
vysledok	11	11	11
x	5	5	5
y	6	6	6

Obr. 4 Prostredie Workspace (Vlastné spracovanie)

### 2.1.6 Command History

V slovenskom preklade „história príkazov“, zobrazuje všetky príkazy, ktoré boli použité v hlavnom okne Command Window. (Kupka, 2007, s. 11)

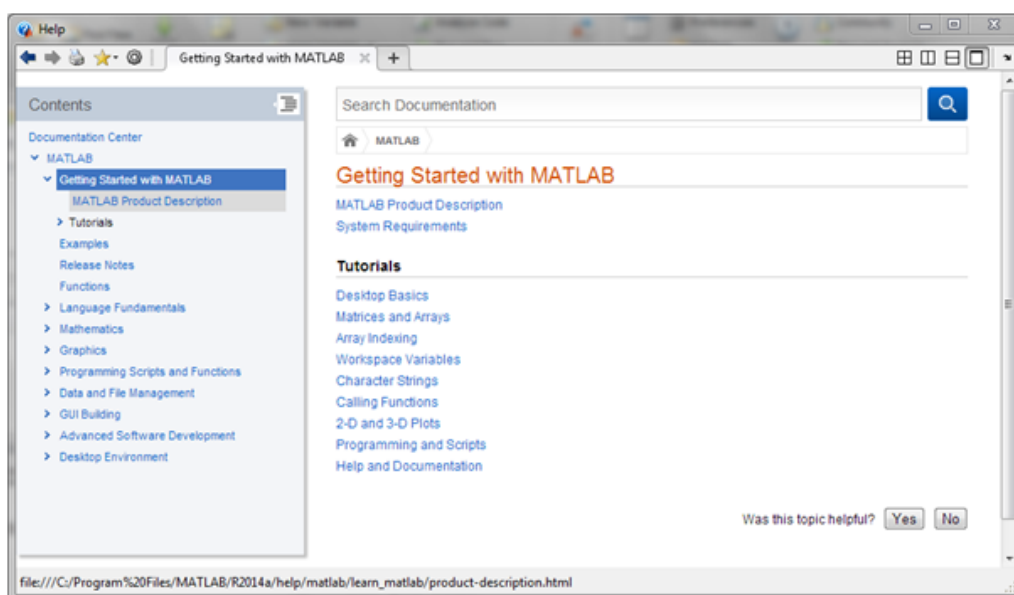
Vypíše sa dátum a čas použitého príkazu a samostatný príkaz, ako je zobrazené na Obr. 5. V prípade, že už bol príkaz raz použitý, je ho možné použiť znova, pretiahnutím do príkazového okna, alebo dvojklikom na daný príkaz.



Obr. 5 Prostredie Command History  
(Vlastné spracovanie)

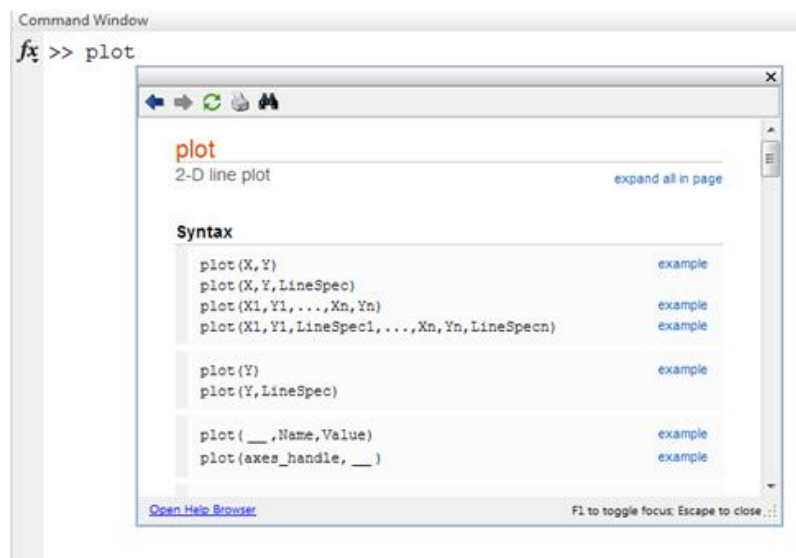
### 2.1.7 Nápoveda

Obsahuje ako veľké množstvo syntaxie funkcií tak aj veľké množstvo príkladov. K zvládnutiu práce s MATLABom je nutné osvojiť si používanie nápovedy. Nápovedu k MATLABu vyvoláme z hlavného menu volaním položky HELP – MATLAB Help alebo pomocou klávesovej skratky F1. Zobrazené okno nápovedy obsahuje množstvo kapitol viz Obr. 6. Prvou kapitolou je MATLAB -> Začínáme s MATLABom [Getting Started with MATLAB]. Táto kapitola je určená pre začiatočníkov, kde sa dozvie o základoch práce s MATLABom. Tu sa nachádzajú základné informácie ako pracovať s maticami, ako kresliť grafy a ako programovať. (Karban, 2006, s. 18)



Obr. 6 Nápoveda programu MATLAB (Vlastné spracovanie)

Pri práci v príkazovom okne je veľmi dobre využiteľná nápoveda vo forme: *help nazov funcie*, alebo napísanie kusu funkcie a použitie klávesy F1. V tomto prípade sa vypíše skrátená syntaxia pre požadovanú funkciu, ako je možné vidieť na Obr. 7.

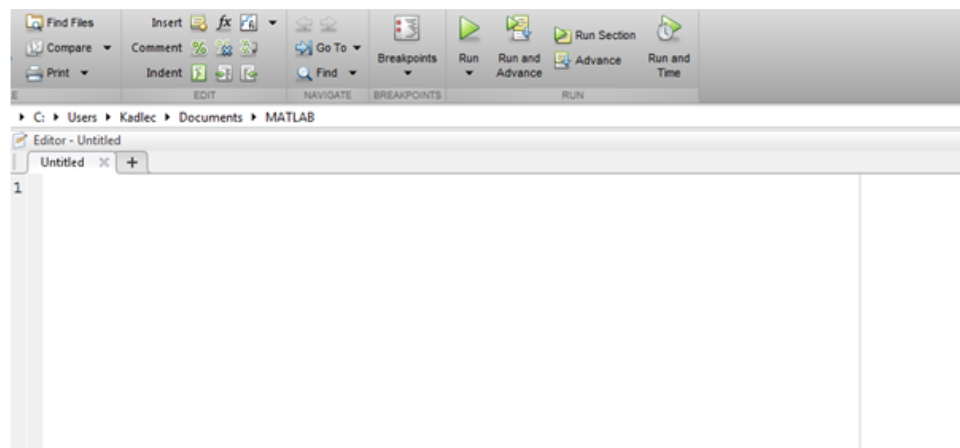


Obr. 7 Syntaxová nápoveda (Vlastné spracovanie)

## 2.2 Vytvorenie skriptu (M-FILES) v MATLABe

V MATLABe je možné jednotlivé príkazy zadávať priamo do príkazového okna alebo vytvoriť M-súbor (m-files). Sú to súbory s príponou \*.m. Obsahujú postupnosť príkazov a povelov systému MATLAB a sú uložené na pevnom disku v počítači, alebo na inom záznamovom médiu. M-súbor je možné vytvoriť nielen v programe MATLAB, ale aj v ľubovoľnom textovom editore, ktorý nepridáva žiadne informácie o type formátovania a ktorý umožňuje zapísať vlastnú príponu \*.m. Pre užívateľa je ale najjednoduchšie použiť editor, ktorý obsahuje MATLAB. M-súbory sú teda zdrojový text či kód aplikácie s ktorým vie MATLAB pracovať. Výhody M-súborov sú celkom veľké. Napríklad aby sa zachovali príkazy v prípade ukončenia práce so systémom, pri výpadku prúdu alebo pri prechode na iný počítač s MATLABom. Pomocou M-súborov je možné taktiež vytvoriť grafické objekty. (Zaplatílek a Doňar, 2004, s. 15)

M-súbor sa po uložení spúšťa pomocou zelenej ikonky spustenia (RUN) v paneli nástrojov, ktorý sa nachádza v hornej časti M-súboru viz Obr. 8.



Obr. 8 M-súbor v MATLABu (Vlastné spracovanie)

## 2.3 Práca v príkazovom riadku

Príkazový riadok (Command Window) je základným oknom prostredia MATLAB. Slúži k zadávaniu príkazov, spusteniu skriptov (M-súborov) a jednotlivých funkcií MATLABu, výpisu premenných a prípadne k spusteniu nápovedy pomocou príkazu help.

Medzi najzákladnejšie príkazy patrí:

- helpwin - zavolanie okna Help pre nápovedu,
- help <funkcia> - nápoveda k zadanej funkcii v okne Command Window,
- doc <funkcie> - podrobnejšia nápoveda k zadanej funkcii v okne Help,
- pwd – výpis aktuálneho adresára vrátane úplnej cesty,
- dir, ls – výpis obsahu aktuálneho adresára,
- format – nastavenie výstupného zobrazovaného formátu v príkazovom okne,
- lookfor – vyhľadávanie zadaného kľúčového slova vo všetkých nápovedách,
- workspace – zobrazenie okna pracovného priestoru,
- cd – zmenenie pracovného adresára alebo vypísanie aktuálneho vrátane cesty,
- mkdir – vytvorenie nového adresára,
- rmdir – odstránenie adresára,
- copyfile – kopírovanie súborov alebo adresárov,
- movefile – presúvanie súborov alebo adresárov,
- what – vracia zoznam špecifických súborov v aktuálnom adresáre,
- type – výpis obsahu M- súboru,
- whos – výpis premenných (textová kópia Workspace),

- who – zjednodušený výpis premenných (iba ich názov),
  - clear <premenná> - clear all – vymazanie premennej resp. všetkých premenných.
- (Kupka, 2007, s. 15)

## 2.4 Typy matlab súborov registrované v MS Windows

V tejto časti budú popísané jednotlivé súbory, ktoré sú registrované v MS Windows.

### 2.4.1 Súbory typu .FIG

Slúži pre uloženie informácií o obsahu grafického okna. Toto okno môže byť dialóg s menu a rôznymi ovládacími prvkami alebo okno s objektom typu axes, v ktorom sú znázornené grafy či obrázky. Súbor s príponou FIG je registrovaný priamo na súbor matlab.exe. Ide o predvolený formát pri ukladaní grafických výstupov a GUI vytvorených pomocou GUIDE. (Perůtka, 2005, s. 29)

### 2.4.2 Súbory typu .M

Sú to textové súbory, do ktorých sa píše kód funkcií a príkazov v poradí, akom by mali byť volané. Pre písanie a úpravu týchto súborov stačí jednoduchý textový editor (napr. Poznámkový blok) Do týchto súborov sa píše vlastné skripty, funkcie, alebo len postupnosti príkazov rovnako ako v Command Windows. Prípona M je registrovaná na medit.exe, teda na Editor v MATLABe, ktorý slúži zároveň ako ladiaci program. (Perůtka, 2005, s. 30)

### 2.4.3 Súbory typu MAT

Ide o súbory, do ktorých MATLAB ukladá dáta na disk MAT- súbory môžu obsahovať ľubovoľný dátový typ podporovaný verziou MATLABu (napr. reťazce, matice, multidimensionálne pole, štruktúry a bunkové pole). MATLAB dáta na disk zapisuje sekvenčne, ako spojitú sekvenciu bajtov. rovnako ako FIG je aj MAT registrovaný priamo na súbor matlab.exe. Ukladanie sa vykonáva pomocou funkcie *save*, čítanie pomocou funkcie *load*. (Perůtka, 2005, s. 30)

### 2.4.4 Súbory typu .P

Tieto súbory sú predspracované pseudokódové súbory. Jedná sa o zakódované súbory zdrojového kódu príponou M. Kódovanie sa vykonáva funkciou *pcode*. Pre dekódovanie, čítanie a úpravu tohto typu súboru ale žiaden príkaz neexistuje. Ale je ho možné v MAT-



LABe spustiť rovnako ako súbor s príponou M. Kodér je samostatný, a preto môže nastať problém s kompatibilitou u starších verzií. Verzia kodéru je uvedená v hlavičke P súboru. (Perůtka, 2005, s. 30)

#### 2.4.5 Súbory typu MEX

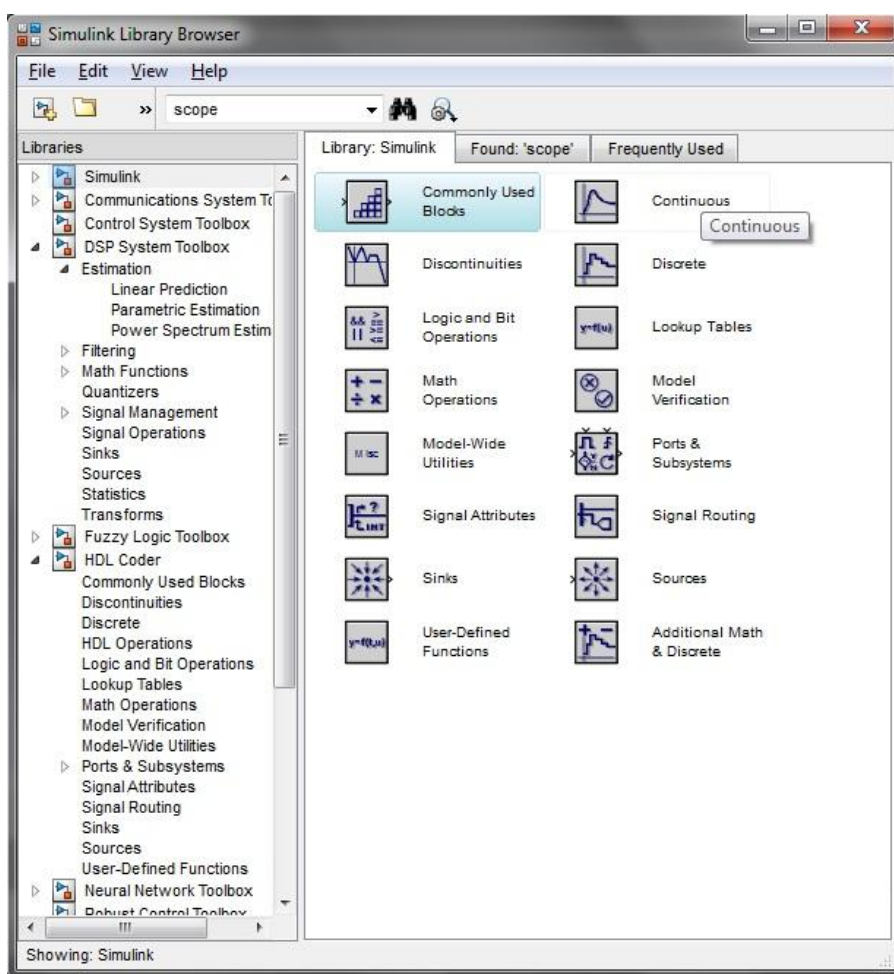
Sú to subrutiny vytvorené zo zdrojového kódu jazyka C alebo Fortran. Ich chovanie je rovnaké ako u M-súborov a vstavaných funkcií, nemajú však pevne danú príponu. Prípona závisí na platforme: napr MS WINDOWS – prípona *dll*, LINUX – prípona *mexglx*. Pokiaľ má M-súbor a MEX súbor rovnaké meno, prednosť má MEX súbor pri volaní v MATLABe. MEX súbory sú plne integrované, takže je možné z príslušného kódu v jazyku C volať funkcie, operátory a ďalšie M-súbory MATLABu. Volanie je realizované pomocou API funkcie *mexCallMATLAB*. (Perůtka, 2005, s. 30)

### 3 POPIS PROGRAMOVACIEHO PROSTREDIA SIMULINK

V tejto kapitole bude vysvetlené prostredie Simulinku

#### 3.1 Spustenie Simulinku

Simulink sa dá spustiť viacerými spôsobmi, medzi najpoužívannejšie patria dva. Prvý spôsob spustenia programu je cez príkazové okno Command Window zapísaním príkazu „*simulink*“. Druhou možnosťou ako otvoriť Simulink je kliknutím na ikonu Simulink Library, ktorá sa nachádza v hornom menu v záložke home viz Obr. 9.



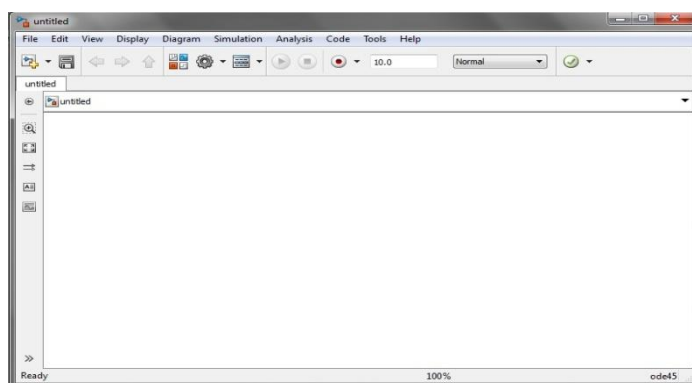
Obr. 9 Zobrazení Simulink Library Browser

(Vlastné spracovanie)

Následne sa otvorí okno z názvom Simulink Library Browser viz Obr. 9, ktoré obsahuje štandardné knižnice. Každá z knižníc obsahuje jednotlivé bloky, reprezentujúce príslušné funkcie. Bloky sú v knižniciach zoskupené podľa oblasti ich použitia. Do mnohých blokov je možné zadať hodnoty rôznych parametrov. Hodnoty je možné do týchto blokov zadávať

priamo, alebo pomocou definovaných premenných, čo je efektívnejšie. Premenné je možné nadefinovať štandardným spôsobom ako v základnom prostredí MATLABu. V tomto prípade je nutné premenné inicializovať pred spustením vlastnej simulácie, a to buď zadaním do príkazového okna v MATLABe, alebo pomocou skriptu, súčasťou môže byť aj prípadné spustenie simulácie a vykreslenie výsledkov v grafickom formáte. (Kupka, 2007, s. 116-117)

Okno grafického prostredia sa otvorí buď cez hlavné menu v MATLABe **New -> Simulink Model**, alebo cez hlavné menu v Simulink Library Browser a to **File -> New -> Model**. Zobrazí sa jednoduché okno, do ktorého sa jednotlivé bloky dajú vložiť z knižníc a tak zostaviť model pre simuláciu.

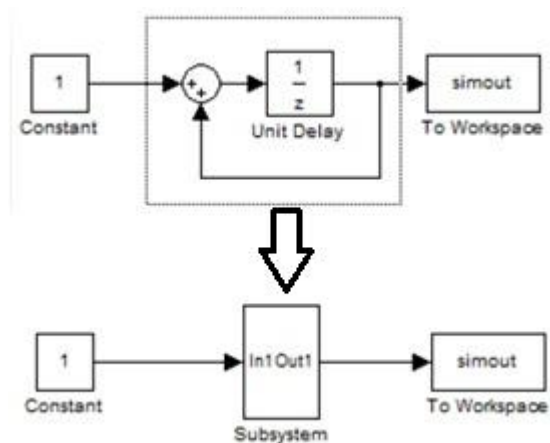


Obr. 10 Grafické prostredie Simulink (Vlastné spracovanie)

### 3.2 Subsystemy a maska subsystemu

Kupka (2007, s. 141-146) uvádza, že subsystem reprezentuje sadu blokov, ktoré môžu byť nahradené jedným blokom. Výhodou subsystemu je, že ako náhle sa model zväčšuje a stáva sa zložitým, môžeme ho jednoducho zoskupiť do subsystemu. Vytvoriť subsystem je možné tak, že sa označujú bloky, ktoré budú zjednotené v subsysteme a po kliknutí pravým tlačidlom na výber bude zvolená možnosť **Create Subsystem**. Toto zjednotenie je možné taktiež vybrať v hornom menu okna modelu v zložke **Edit** (alebo **Ctrl + G**). Označené bloky sa okamžite zjednotia do subsystemu. Novo vzniknutý blok má názov **Subsystem** a jeho ikona obsahuje aj popis vstupov a výstupov. Kliknutím pravého tlačidla na subsystem a zvolením možnosti **Mask Subsystem** sa vytvorí maska subsystemu. Túto možnosť je taktiež možné zvoliť v hornom menu okna modelu v záložke **Edit -> Mask Subsystem** (alebo **Ctrl + M**). Pomocou masky je možné meniť vzhľad bloku, parametre bloku vo vnútri masky a je možné vytvoriť aj dokumentáciu alebo nápoved' pre maskovaný blok.

Subsystémy je vidieť na Obr. 11.



Obr. 11 *Subsystem (Vlastné spracovanie)*

## 4 PREMENNÉ

Pre jednoduchý kód v MATLABe nie je potreba definovať ani deklarovať premenné (pokiaľ nejde o globálne alebo persistent premenné). MATLAB definovanie a deklaráciu vykoná sám pri prvom použití a uloží ich do základného pracovného priestoru. Jedná sa o premenné, ktoré sa objavujú pri zadávaní príkazov v Command Window MATLABu a taktiež premenné skriptov (M-súborov), ktoré je možné kopírovať a spúšťať po riadkoch v ich poradí. (Perůtka, 2005, s. 33)

Pri použití funkcií sa premenné ukladajú do lokálneho priestoru danej funkcie, čo zvyšuje výpočtovú rýchlosť. Tento lokálny priestor funkcie je oddelený od ostatných lokálnych priestorov iných funkcií, a taktiež je oddelený od základného pracovného priestoru MATLABu. Znamená to, že je možné použiť napríklad premennú rovnakého názvu v dvoch funkciách v jednom skripte a pritom ich nie je možné prepísať. Pokiaľ je potrebné teda zistiť hodnotu určitej premennej vo vnútri funkcie, je potreba túto premennú použiť ako argument výstupu danej funkcie. Ak je potreba premennú upraviť, musí sa použiť ako vstupný a výstupný argument funkcie. Tento postup sprehľadňuje a zrýchľuje chod programu. Premenné uložené v lokálnom priestore danej funkcie sa automaticky po ukončení funkcie vymažú, zatiaľ čo u premenných, ktoré je možné zadať priamo do Command Window, zostávajú a ich zoznam je možné zavolať pomocou príkazu `who`, alebo podrobnejšie pomocou príkazu `whos`. (Perůtka, 2005, s. 33)

Ďalšou možnosťou je definovať premennú ako globálnu, čím sa uloží do globálneho priestoru. Hodnota tejto globálnej premennej sa dá potom meniť vo všetkých funkciách a skriptoch.

Premennú typu *persistent* je možné čítať a meniť rodičovskou funkciou, u tohto typu premennej je potreba použiť ešte 3 ďalšie funkcie pre správnu obsluhu.

MATLAB rozlišuje malé a veľké písmena, je teda rozdiel medzi premennou *cislo* a *Cislo*, ide o celkom dve rozdielne premenné.

Premenná môže mať maximálne 31 znakov, nesmie obsahovať bodku ani diakritiku a vždy musí začínať písmenom. (Kupka, 2005, s. 17)

Tab. 1 Najpoužívanéjšie riadiace znaky (Kupka, 2007, s. 18)

Znak	Popis
. (bodka)	desatinná čiarka, oddelovač premenných v štruktúre
, (čiarka)	oddelovač príkazov na riadku (výpis nie je potlačený) alebo prvkov
: (dvojbodka)	rozsah hodnôt alebo indexov vo vektore alebo matici
; (stredník)	potlačenie výpisu pre prí priradenie do premennej, volanie funkcie alebo M-súboru, oddelovač prvkov v stĺpcoch či celých riadkoch u
% (percento)	oddelenie komentára na riadku
... (tri bodky)	rozdelenie dlhého príkazu
!	spustenie systémového príkazu
[ ]	ohraničenie obsahu definovaného vektoru alebo matice
{ }	ohraničenie obsahu definovanej bunky
( )	použitie pri indexovaní vektorov alebo matic, obsahy
' (apostrof)	ohraničenie textovej premennej, transpozície matice

#### 4.1 Lokálne premenné

Lokálna premenná je predvolený formát premenných používaných MATLABom v Command Window, skriptoch a metódach. Nie je potrebné ju definovať ani deklarovať, všetko je vykonané automaticky pri prvom volaní premennej. U číselných premenných je predvoleným formátom formát *double*. Pokiaľ je potreba upraviť rýchlosť programu, môžeme číselné premenné deklarovať na začiatku skriptu, napríklad pomocou príkazu *zeros*. (Perůtka, 2005, s. 33)

Príklad pretypovania lokálnej premennej z *double* na *string*:

- `LokPremennaDouble = 10;`
- `LokPremennaString = int2str(LokPremennaDouble)`

#### 4.2 Globálne premenné

Premenná *global* sa používa kvôli funkciám. Každá funkcia MATLABu, ktorá je definovaná v M-súbore má svoje vlastné lokálne premenné oddelené od ostatných premenných pracovného priestoru a ostatných funkcií. Je však možné zdieľať kópiu jednej premennej

v pracovnom priestore aj u niekoľkých funkcií naraz. Je treba použiť príkaz *global* a premennú deklarovať. Všetky funkcie, ktoré ju majú deklarovanú ako globálnu, ju môžu používať a meniť jej obsah. (Perůtka, 2005, s. 34)

Vymazať globálnu premennú je možné príkazom *clear premenna*, kde *premenna* je názov globálnej premennej typu *global*. Týmto sa premenná odstráni z hlavného pracovného priestoru MATLABu, ale hodnota globálnej premennej sa nezmení.

Pre odstránenie globálnej premennej z globálneho priestoru globálnych premenných je potreba zadať príkaz *clear global premenna*, kde *premenna* je názov globálnej premennej.

Globálne premenné sa kvôli odlíšeniu od lokálnych píšú tak, že všetky ich znaky sú kapi-tálkami. (Perůtka, 2005, s. 34)

### 4.3 Presistentné premenné

Tieto premenné je možné používať iba v jednej funkcii, ostatné funkcie k nej nemajú priamy prístup, okrem tej, v ktorej bola deklarovaná. Tým sa zaručí, že nebude prepísaná. Po skončení funkcie táto premenná nie je vymazaná. Ak sa funkcia, v ktorej je táto premenná, odstráni z pamäti alebo edituje M-súbor tejto funkcie, zmažú sa všetky premenné typu *persistent* tejto funkcie. Premennú je potreba deklarovať pred jej použitím priamo vo funkcii. Deklarujeme ju pomocou príkazu *persistent premenna*, kde *premenna* je názov persistentnej premennej. Obvykle sa deklaruje na začiatku rovnako ako premenná *global* a taktiež i tu sa píšú všetky znaky kapitálkami pre odlíšenie od lokálnych premenných. (Perůtka, 2005, s. 35)

### 4.4 Rezervované kľúčové slová

Tieto slová nie je možné použiť ako názvy premenných:

*Ans, beep, bitmax, break, case, catch, continue, else, elseif, end, for, function, global, i, if, Inf, j, NaN, nargin, nargout, otherwise, pi, persistent, realmin, realmax, return, switch, try, varargin, varargout, while.* (Perůtka, 2005, s.35)

## 5 DATOVÉ TYPY

MATLAB obsahuje 15 základných dátových typov a každý z nich môže byť zapísaný vo forme matice alebo poľa. Všetky základné typy sú zobrazené na obrázku. (Karban, 2006, s. 18-19)

### 5.1 Celočíselné dátové typy

Dátové typy, ktoré pracujú s číslami je možné rozdeliť do dvoch základných skupín.

- a) Pracujú s celými číslami:
  - kladné čísla (*uint8*, *uint16*, *uint32*, *uint64*),
  - záporné čísla (*int8*, *int16*, *int32*, *int64*).
- b) Pracujú s reálnymi číslami:
  - *single*, *double*. (Perůtka, 2005, s. 36)

### 5.2 Číselný dátový typ *double*

Je to predvolený a najčastejšie používaný dátový typ v MATLABe. Číslo uložené v tomto type premennej je zložené z dvoch častí, a to z celej a desatinnej časti, kde počet desatinných miest nie je pevne daný. Počet desatinných miest sa môže meniť v závislosti na požadovanej presnosti.

K pretypovaniu čísla z jedného dátového typu do typu *double* sa používa funkcia *double*. (Perůtka, 2005, s. 36)

### 5.3 Číselný dátový typ *single*

Premenné tohto typu majú podobné vlastnosti ako premenné typu *double*, rozdiel je iba v nižšej presnosti a rozsahu. Vďaka tomu, ale tieto premenné zaberajú menej pamäťového priestoru. Ich použitie je možné, pokiaľ to situácia dovoľuje, následkom toho je v časovo náročných výpočtoch umožnený rýchlejší chod programu.

V súčasných verziách MATLABu sú už definované matematické operácie aj pre tento dátový typ. (Perůtka, 2005, s. 37-37)



## 5.4 Dátové typy uint8, uint16, uint32 a uint64

Tieto dátové typy môžu byť čísla v rozsahu 0 až 2 na mocninu čísla uvedeného u daného dátového typu mínus 1. Sú to čísla 8, 16, 32, 64. Tieto čísla zároveň udávajú počet bitov daného dátového typu viz Tab. 2.

Tab. 2 Dátové typy uint8, uint16, uint32, uint64  
(MathWorks, ©1994-2015)

Function	Output Range	Output type	Bytes per Element	Output Class
uint8	0 to 255	Unsigned 8-bit integer	1	uint8
uint16	0 to 65,535	Unsigned 16-bit integer	2	uint16
uint32	0 to 4,294,967,295	Unsigned 32-bit integer	4	uint32
uint64	0 to 18,446,744,073,709,551,615	Unsigned 64-bit integer	8	uint64

V súčasných verziách MATLABu sú už definované matematické operácie aj pre tieto dátové typy. Pokiaľ dôjde k prekročeniu hranice daného typu výsledkom je maximum (napr. pre uint8 je to 255).

Pretypovanie – príklad:

```
x=cislo;                                % Premenná je typu double – t.j. predvolený
                                         typ premennej
i = uint8(x)
i = uint16(x)
i = uint32(x)
i = uint64(x)
whos                                     % Vypíše premenné aj s dátovými typmi
```

## 5.5 Dátové typy int8, int16, int32, int64

Tieto dátové typy majú podobný rozsah ako predchádzajúce typy unit, ale na rozdiel od dátového typu unit, môžu ísť aj do záporných čísel a ich maximálna hodnota je o polovicu menšia ako u dátového typu unit. Čísla udané za označením dátového typu udávajú počet bitov daného dátového typu viz Tab. 3.

Tab. 3 Dátové typy *int8*, *int16*, *int32*, *int64*

(MathWorks, ©1994-2015)

Function	Output Range	Output type	Bytes per Element	Output Class
<i>int8</i>	(-128 to 127)	Signed 8-bit integer	1	<i>int8</i>
<i>int16</i>	(-32,768 to 32,767)	Signed 16-bit integer	2	<i>int16</i>
<i>int32</i>	(-2,147,483,648 to 2,147,483,647)	Signed 32-bit integer	4	<i>int32</i>
<i>int64</i>	(-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)	Signed 64-bit integer	8	<i>int64</i>

V súčasných verziách MATLABu sú už definované matematické operácie aj pre tieto dátové typy. Pokiaľ dôjde k prekročeniu hranice daného typu výsledkom je maximum (napr. pre *int8* je to 127).

Pretypovanie – príklad:

```

x=cislo;                                     % Premenná je typu double – t.j. predvolený
                                              typ premennej
int8(x)
int16(x)
int32(x)
int64(x)
whos                                         % Vypíše premenné i s dátovými typmi

```

## 5.6 Dátový typ char

Tento dátový typ je určený pre znaky a reťazce. Reťazce typu *char* sú 16-bitové reťazce. Rozsah tohto dátového typu je rovnaký ako u *uint16*, teda 0-65535. Reťazce sú ukončené prvkom NULL. (Perůtka, 2005, s. 38)

Pretypovanie

Príklad, v ktorom sa deklaruje premenná typu *double* a pomocou príkazu *char* sa vypíše znak z ASCII tabuľky, ktorý odpovedá danej hodnote, ktorá je uložená v premennej *cislo*.

Pretypovanie – príklad:

```

cislo=65                                     % Premenná cislo je dátového typu double
char(cislo)

ans = A          - ans je typu char

```

## 5.7 Dátový typ Cell

Tento dátový typ sa používa ako bunečné pole – cell array. Tento formát je špecifický len pre MATLAB. Jedná sa o pole, ktoré je podobné matici, alebo vektoru. (Blaho, 2007) Základným rozdielom je, že prvkami v matici sú čísla, ale v bunke môže byť ľubovoľný dátový typ. Bunka môže obsahovať číslo, vektor, štruktúru, text a dokonca aj ďalšiu bunku. Môže tak vzniknúť komplikovaný dátový objekt s veľkým množstvom informácií rôzneho typu. Bunka dátového typu cell môže mať podobne ako matica viac rozmerov. Na rozdiel od matic sa jednotlivé prvky adresujú v zložených zátvorkách. (*MathWorks*, ©1994-2015)

### Deklarovanie Cell

Deklarovanie premennej, ktorá je z triedy *cell* je možné dvomi spôsobmi. Prvým spôsobom je využitie príkazu *cell* s parametrami, ktoré predstavujú rozmer bunky. Bunku s menom A a rozmerom 2×2 je možné deklarovať nasledovne:

```
>> A=cell(2,2)
A =
[] []
[] []
>> whos
Name Size Bytes Class Attributes
A 2x2 16 cell
```

V tomto okamihu máme k dispozícii bunku A, ktorá je prázdna.

Druhý spôsob je pre MATLAB typický. Jedná sa o deklarovanie priradením. Nie je ani v tomto prípade explicitná deklarácia. Postačuje prvé priradenie a MATLAB okamžite vyhradí pamäť pre zvolený typ. Hodnoty, ktoré sa priradujú do bunky sú uzavreté v zložených zátvorkách. (Blaho, 2007)

```
>> B={[] [1 2;3 4]; 'Posterus' [4 5 6] }
B =
[] [2x2 double]
'Posterus' [1x3 double]
>> whos
Name Size Bytes Class Attributes
A 2x2 80 cell
B 2x2 320 cell
```

Tento príklad demonštruje variabilitu triedy cell. Bunka B obsahuje skalár, maticu, text a vektor.

Bunky sa dajú plniť naraz, alebo postupne jednotlivými položkami.

```
>> A(1,1)={1};  
>> A(1,2)={'textovy retazec'};  
>> A(2,1)={magic(4)}  
A =  
[ 1] 'textovy retazec'  
[4x4 double] []
```

## 5.8 Dátový typ struct

Ide o dátový typ, ktorý definuje v sebe pole premenných rôzneho typu. Každá položka pola musí mať presne definovaný typ a nie je ich možné kombinovať. V rámci pola sa však môžu kombinovať dátové typy ľubovoľne. Každé pole štruktúry má svoje meno, tzv. textový menovateľ pola.

V MATLABe sú dve možnosti ako štruktúru vytvoriť. Prvým z nich je jednoduché priradenie do jednotlivých položiek. Položka sa od názvu štruktúry oddeľuje bodkou. (Blaho, 2007)

*štruktúra.položka = hodnota\_položky*

Príklad:

```
>> auto.znacka='MUSTANG';  
>> auto.model='Shelby';  
>> auto.rok_vyroby=1967;
```

Výsledok:

```
>> auto  
auto =  
znacka: 'MUSTANG'  
model: 'Shelby'  
rok_vyroby: 1967
```

Pomocou príkazu whos sa zisťuje koľko vytvorená štruktúra zaberá miesta v pamäti a ako ju MATLAB vníma.

```
>> whos  
Name Size Bytes Class Attributes  
kniha 1x1 520 struct
```

Ďalšie pole sa pridá takto:

```
>> auto(2).znacka='Mitsubishi';  
>> auto(2).model='Lancer';  
>> auto(2).rok_vyroby=2010;
```

Druhý spôsob ako definovať štruktúru je využiť príkaz `struct`. Syntax príkazu je:

```
s = struct('field1',VALUES1,'field2',VALUES2,...)
```

V tomto príklade to znamená:

```
auto(2)=struct('znacka','Mitsubishi','model','Lancer','rok_vyroby','2010');
```

## 5.9 Dátový typ Handle function @

*Handle function* je dátový typ MATLABu, ktorý obsahuje informácie pre odkazovanie sa na funkciu. Po vytvorení handle funkcie MATLAB do handle uloží všetky informácie potrebné k tomu, aby sa funkcia spustila alebo vyhodnotila pomocou funkcie *feval*. Handle function je viac než len odkaz na danú funkciu. Často reprezentuje sadu metód funkcie. Keď vyhodnocujeme handle function, MATLAB berie do úvahy iba funkcie, ktoré boli uložené do handle, keď bol vytvorený. S ostatnými funkciami, ktoré sa môžu nachádzať v cestách MATLABu nie je počítané.

Výhody handle function:

- umožňuje prístup k funkcii inými funkciami,
- zachytáva všetky metódy preťaženej funkcie,
- umožňuje širší prístup k subfunkciám a privátnym funkciam,
- zvyšuje prehľadnosť,
- znižuje počet súborov,
- zvyšuje výkon programu pri opakovaných operáciách. (Perůtka, 2005, s. 42)

Handle patrí k štandardným dátovým typom MATLABu. Je s ním možné vykonávať operácie ako s ostatnými dátovými typmi, tj. môže sa vytvárať pole, štruktúry alebo bunkové polia handlu funkcií. Prístup k dátam je rovnaký ako u numerických dátových typov. (Perůtka, 2005, s. 42), (Blaho, 2007)

Handle funkcia sa vytvára tak, že pred názov funkcie sa vloží znak @ a následne sa priradí do premennej.

Príklad:

```
handleSinus=@sin
feval(handleSinus,0.23)
ans =
0.2280
```

## 5.10 Komplexné čísla

Komplexné čísla pozostávajú z dvoch častí: reálnej časti a imaginárnej časti. Základná imaginárna jednotka je rovná odmocnine z čísla -1. MATLAB reprezentuje imaginárnu jednotku dvoma písmenami *i* a *j*.

Komplexné číslo sa dá v MATLABe vytvoriť dvoma spôsobmi. Prvý spôsob je pomocou priameho priradenia do premennej. Druhý spôsob využíva príkaz *complex*, do ktorého sa ako prvý argument zadáva reálna časť a ako druhý argument imaginárna časť komplexného čísla. (Blaho, 2007)

```
>>x = 1 + 1i;  
>>y = complex(1,1);
```

Z komplexného čísla je možné osamostatniť reálnu a imaginárnu časť s využitím funkcií *real* a *imag*.

```
>>xreal = real(x)  
xreal = 1  
>>ximag = imag(x);  
ximag = 1
```

Na vyhodnotenie, či je dané číslo imaginárne alebo len reálne, je možné využiť funkciu *isreal*, ktorá vracia logickú premennú 1 alebo 0. Komplexné číslo s nulovou imaginárnou časťou sa taktiež považuje za reálne. Treba si dávať pozor nato, že časť získaná príkazom *imag* už nie je komplexné číslo. (Blaho, 2007)

```
>>isreal(1+0i);  
ans= 1  
>>isreal(ximag);  
ans = 1
```

## 6 OPERÁTORY

Operátory sa v MATLABe rozdeľujú do troch základných skupín:

- aritmetické,
- relačné,
- logické. (UIAM FCHPT STU, ©2015)

### 6.1 Aritmetické operátory

Aritmetické operátory sa rozdeľujú do dvoch skupín, a to na unárne viz Tab. 4 a binárne viz Tab. 5.

Unárne

Tab. 4 *Unárne aritmetické operátory*  
(UIAM FCHPT STU, ©2015)

Názov	Syntax	Opis
unárny plus	(+a1)	
unárny mínus	(-a1)	
transpozícia a konjugovanosť	(a1')	
transpozícia	(a1.')	

Binárne

Tab. 5 *Binárne aritmetické operátor*  
(UIAM FCHPT STU, ©2015)

Názov	Syntax	Opis
sčítanie	(a1+a2)	
odčítanie	(a1-a2)	
nasobenie matic	(a1*a2)	
nasobenie po prvkoch	(a1.*a2)	
omocnenie matice	(a1^a2)	
umocnenie po prvkoch	(a1.^a2)	
delenie matic	(a1/a2)	
delenie po prvkoch	(a1./a2)	
delenie matic zľava	(a1\ a2)	
delenie po prvkoch zľava	(a1.\ a2)	

## 6.2 Relačné operátory

Tieto operátory slúžia k porovnaniu dvoch hodnôt (premenných, polí rovnakej dimenzie a dĺžky atď.). Výsledkom je číslo logického typu dát tj. logická 0 = false, alebo logická 1 = true. (Perůtka, 2005, s. 44)

Relačné operátory sú:

- <      menší než,
- >      väčší než,
- ==     rovná sa,
- <=    menší než alebo rovná sa,
- >=    väčší než alebo rovná sa,
- ~=     nerovná sa. (UIAM FCHPT STU, ©2015)

## 6.3 Logické operátory

Sú to operátory, ktoré sa aplikujú na jednotlivé prvky, polia (po prvkoch), tj. skaláry.

Logické operátory sú:

- ~      logická negácia – not, pre pole,
- &      logický súčin – and, pre pole,
- |      logický súčet – or, pre pole,
- &&    logický súčin – and, pre podmienky, pre skalár,
- ||     logický súčet – or, pre podmienky, pre skalár. (Perůtka, 2005, s. 45)

Výsledkom pri použití logických operátorov je logická 0 = *false*, alebo logická 1 = *true*.



## 6.4 Špeciálne znaky

Tab. 6 Špeciálne znaky (UIAM FCHPT STU, ©2015)

Znak	Opis	Znak	Opis
+	plus	...	pokračovanie
-	minus	;	bodkočiarka
*	maticové násobenie	,	čiarka
.*	násobenie prvkov	%	komentár
^	umocnenie	!	výkričník
.^	umocnenie prvkov	'	transpozícia
Kron	násobenie Kronecker	.'	
\	ľavé delenie	=	priradenia
/	pravé delenie	==	zhidnosť
./	pravé delenie prvkov	<>	relačné operátory
:	dvojbodka	&	logický AND
[]	hranaté zátvorky		logický OR
()	okružle zátvorky	~	logická negácia
.	desetinná bodka	XOR	log. Exclusive OR
..	rodičovský adresár		

### 6.4.1 Priorita operátorov

Tab. 7 Priorita operátorov (UIAM FCHPT STU, ©2015)

Priorita	Operátor	Poznámka
1.	()	zátvorky
2.	.' , .^ , ^ , .^	traspozícia, umocnenie, transpozícia + konjugovanosť, maticové umocnenie,
3.	~, +, -	unárny plus, unárny minus, negácia
4.	.*, ./, \, *, /, \	násobenie po prvkoch, delenie po prvkoch zľava, násobenie matic, delenie matic, delenie matic zľava
5.	(+, -)	sčítanie, odčítanie
6.	:	dvojbodka
7.	<, <=, >, >=, ==, ~=	relačné operátor
8.	&	logický súčin, AND
9.		logický súčet, OR

## 7 PODMIENKY A CYKLY

V tejto kapitole budú vysvetlené jednotlivé podmienky a cykly, ktoré sú dôležitou súčasťou pri práci v programe MATLAB.

### 7.1 Podmienka If

Podmienka *if* má nasledujúcu syntaxiu:

```
if podmienka1  
    príkazy1  
elseif podmienka2  
    príkazy2  
else  
    príkazy3  
end
```

Príkaz *if* vyhodnocuje splnenie podmienky1. Ak je splnená, vykoná príkazy1. Ak podmienka1 nie je splnená prejde sa do vetvy *elseif* a vyhodnotí sa pravdivosť podmienky2. Ak je podmienka2 splnená vykonajú sa príkazy2. Ak by ani jedna z podmienok nebola pravdivá vykonáva sa vetva *else*, teda príkazy3. Slovo *end* ukončuje príkaz vetvenia. (Blaho, 2007)

Príklad využitia podmienky *if*:

```
if a>b  
    disp('Vacsia je premenna a')  
elseif a<b  
    disp('Vacsia je premenna b')  
else  
    disp('Premenne su rovnake')  
end
```

Vetva *if* môže byť len jedna. Vetva *elseif* nie je povinná, ale naopak ju je možné použiť aj niekoľkokrát, podľa toho koľko podmienok je potreba. Vetva *else* taktiež nie je povinná, no v prípade potreby ju je možné využiť len jeden krát. (Blaho, 2007)

### 7.2 Mnohonásobné vetvenie switch

Podmienka *switch* má nasledujúcu syntaxiu:

```
switch výraz  
    case hodnota1  
        príkazy1  
    case hodnota2  
        príkazy2
```

```
otherwise  
    příkazy3  
end
```

Výraz za příkazom *switch* je vyhodnotený a porovnávaný na hodnoty za slovom *case*. Pokiaľ nastane zhoda (log 1) vykonajú sa príkazy, ktoré obsahuje príslušný *case*. Slovo *otherwise* hovorí čo sa má vykonať keď nie je žiadna hodnota (log 0) zodpovedajúca výrazu. (Perůtka, 2005, s. 47-78) MATLAB končí prehľadávanie keď nájde hľadanú hodnotu preto sa na rozdiel od iných programovacích jazykov nepoužíva kľúčové slovo *break* pri používaní príkazu *switch*. Výrazom za slovom *switch* môže byť reťazec alebo číslo ľubovoľného numerického dátového typu. (Blaho, 2007)

Príklad využitia *switch*:

```
switch farba  
    case cervena  
        disp('farba je cervena')  
    case zelena  
        disp('farba je zelena')  
    case modra  
        disp('farba je modra')  
    otherwise  
        disp('neznáma farba')  
end
```

## 7.3 Cykly

Cyklus slúži pre zápis príkazov, ktoré sa majú vykonávať opakovane. (niekoľko krát za sebou). Počet opakovaní jednotlivých príkazov závisí na určitej podmienke alebo môže byť vopred známy. (Majerova, 2013)

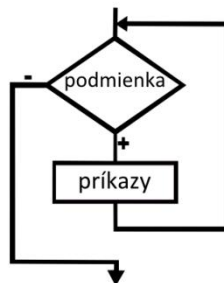
### 7.3.1 Cyklus while

Cyklus *while* má nasledujúcu syntaxiu:

```
while podmienka  
    příkazy  
end
```

Cyklus *while* viz Obr. 12, opakuje príkazy pokým je splnená podmienka. Na začiatku sa testuje platnosť riadiacej podmienky. Pokiaľ podmienka platí, tj. výraz nemá nulovú hodnotu, vykonajú sa všetky príkazy tela cyklu. V ďalšom kroku sa opäť testuje podmienka a pokiaľ platí, vykonajú sa opäť všetky príkazy, ktoré telo cyklu obsahuje. Opakovanie

prebieha dovtedy, kým sa pri testovaní podmienky nezistí, že podmienka už neplatí (tj. hodnota výrazu je nulová). (Majerova, 2013)



Obr. 12 Cyklus *while*

(*Vlastné spracovanie*)

Koniec cyklu *while* môže nastať aj pri prvom testovaní podmienky, je tiež možné, že príkazy v tele cyklu sa nevykonajú ani raz .

Pri nevhodných podmienkach môže nastať nekonečný cyklus. Nekonečný cyklus sa dá ukončiť pomocou klávesovej skratky CTRL+C. (Blaho, 2007)

Príklad využitia *while*:

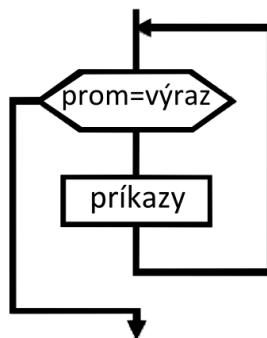
```
i=0;
while i<10
    i=i+1;
end
```

### 7.3.2 Cyklus *for*

Je to cyklus s vopred známym počtom opakovaní. Označuje sa tiež ako iteračný cyklus. Využíva sa teda v prípade, kde je vopred známy počet opakovaní príkazov v tele tohto cyklu. Počet opakovaní je daný vektorom, z ktorého si v každej iterácii (v každom prechode cyklom) zoberieme jednu hodnotu. Táto hodnota je uložená v tzv. riadiacej premennej cyklu. (Majerova, 2013)

Cyklus *for* má nasledujúcu syntaxiu:

```
for premenná = štart : prírastok : koniec
    príkazy
end
```



Obr. 13 Cyklus *for*  
(Vlastné spracovanie)

Pre predčasné ukončenie cyklu je možné použiť príkaz `break`, ktorý môže byť vo vnútri cyklu v podmienke `if`. Výraz môže mať ľubovoľný vektor čísel, nesmie byť však komplexný. (Majerova, 2013)

```

index=0;
for i=0:0.1:10,
    t(index)=I;
    sinus(index)=sin(i);
end
plot(t,sinus) % vykreslenie
  
```

### 7.3.3 Príkazy `continue`, `break` a `return`

Riadenie behu programu je možné ovplyvniť príkazmi `continue`, `break` a `return`. Každý z príkazov má svoje špecifické vlastnosti. Príkaz `continue` preskakuje aktuálnu iteráciu cyklu `for` alebo `while`. Príkazy za `continue` sa už nevykonávajú a pokračuje sa ďalším krokom cyklu. Príkaz `break` slúži na ukončenie vykonávaného cyklu `for` alebo `while`. Vykonávanie programu tak pokračuje ďalej za koncom týchto cyklov. Ak je cyklov vnorených viac ukončuje sa len ten v ktorom sa príkaz `break` nachádza. Príkaz `return` ukončuje aktuálnu postupnosť príkazov a vracia riadenie tomu, kto vyvolal aktuálny program alebo skript, prípadne klávesnici. Týmto príkazom sa ukončí činnosť skôr ako nastane jej bežný koniec o úroveň vyššie. (Blaho, 2007)

## 7.4 Polia, vektory a matice

MATLAB je navrhnutý tak, že sa u neho uvažuje s maximálnym využitím polí, matíc a vektorov. Pole je v MATLABe najdôležitejší „stavebný kameň“, pretože matice, vektor a skalár sú jeho špeciálne prípady. Vyplýva to zo spôsobu ich uloženia v pamäti. Pole je teda

sada čísel, ktoré sa nazývajú elementy poľa, na ktoré sa odkazujeme jedným alebo viacerými indexmi rôznych množín indexov. Dimenzia poľa je počet indexov potrebných pre špecifikáciu elementov poľa. Veľkosť poľa je zoznam veľkostí množín indexov. Dimenzia poľa je počet indexov potrebných pre špecifikáciu elementov poľa. (PERUTKA, 2005, s.51). Matica je teda väčšinou obdĺžnikové pole čísel, u ktorých je dôležitá ich pozícia v poli. Veľkosť matíc je popísaná počtom ich riadkov a stĺpcov. Matica s  $m$  riadkami a  $n$  stĺpcami sa nazýva matica  $m \times n$ . Prvky matice sú indexované dvojitém indexom, ktorý odpovedá číslu ich riadku a stĺpca. (Karban, 2006, s. 40)

## 7.5 Skalár

Skalár je špeciálny prípad matice o rozmere  $1 \times 1$ . Vektor je matica o jednom riadku alebo stĺpci

Príklad:

```
pole = [1 2 3 4 5 6 7 8 9]
matice = [1 2 3; 4 5 6; 7 8 9]
vektor = 0:0.5:10
```

Často používané príkazy pre prácu s maticami:

*size* – veľkosť každej dimenzie,

*length* – veľkosť najdlhšej dimenzie,

*ndims* – počet dimenzií,

*find* – vrátenie indexov nenulových prvkov,

*eye* – jednotková matica,

*ones* – matica jednotiek,

*zeros* – matica nul,

*diag* – diagonálna matica,

*triu* – horní trojuholníková matica,

*tril* – dolná trojuholníková matica,

*rand*, *randn* – náhodné čísla.

## 7.6 Funkcia

Funkcia je M-súbor, ktorý začína slovom `function` a jej názvom je rovnaký názov M-súboru. Je potrebné si dávať pozor, aby sa meno nezačínalo na číslo a iné nepovolené znaky, prípadne sa nezhodovalo s menom inej funkcie v MATLABe. M-súbor. Funkcia má voči skriptu mnoho výhod. Pri zavolaní sa spracuje, skompiluje celá naraz. Môže mať vstupné a výstupné parametre. Premenné funkcie sa ukladajú do pracovného priestoru tejto funkcie a sú tak chránené proti zásahu z iných funkcií, skriptov či Command Window. Po ukončení funkcie sa tieto lokálne premenné odstránia z pamäti. Ak voláme funkciu z Command Window alebo z iného M-súboru, funkcia je rozobraná na príkazy a uložená v pamäti MATLABu do doby, kým nie je zadáný príkaz `clear` alebo do ukončenia MATLABu. (Blaho, 2007)

Príklad:

```
function [A1,A2] = rozdel(A,N)  
%Funkcia rozdeli vektor na dva  
%rozdel(A,N) vracia dva vektory, ktoré vzniknú  
%rozdelením vektora A na pozíci N  
    A1=A(1:N);  
    A2=A(N+1:length(A))  
end
```

Prvý riadok M-súboru hovorí MATLABu, že obsahuje funkciu. Začína sa kľúčovým slovom `function`, nasledujú výstupné argumenty, meno funkcie a vstupné argumenty. Výstupné argumenty sa zadávajú v hranatých zátvorkách, pri jednom výstupnom argumente zátvorky netreba písať. Ak by funkcia nevracala žiadne hodnoty, stačí túto časť vynechať alebo napísať prázdne hranaté zátvorky. Meno funkcie sa musí začínať písmenom, nesmie obsahovať nepovolené znaky, prekročiť povolenú dĺžku a nesmie sa zhodovať s nejakým kľúčovým slovom v MATLABe.

Typy funkcií v MATLABe:

- anonymné funkcie – funkcia v príkazovom riadku,
- primárna funkcia – prvá funkcia v M-súbore,
- subfunkcie – funkcie za primárnou funkciou,
- vnorené funkcie – funkcie v ďalšej funkcii,
- súkromné funkcie – funkcie pracovného adresára,
- preťažené funkcie – funkcie s rovnakým menom a rôznymi argumentmi. (Blaho, 2007)

## **II. PRAKTICKÁ ČÁST**



## 8 ZÁKLADNÉ OPERÁCIE S VEKTORMI A MATICAMI

Praktická časť bude zameraná na príkazy. Bude sa zaoberať jednoduchými príkazmi, ktoré je možné zadávať do Command Window, až zložitejšími skriptami (M-súbory), ktoré budú obsahovať funkcie, zložené z viacerých príkazov, cyklov a podmienok.

V tejto kapitole budú ďalej rozobrané vektory a matice, ktoré patria k jedným z najčastejšie používaným v MATLABe.

### 8.1 Vektory

Vektor je možné zapísať viacerými spôsobmi. Na oddelenie jednotlivých čísel od seba je možné využiť len medzeru, alebo čiarku.

#### Príklad 1: Vytvorenie vektora

```
>>v = [1 2 3]           % Oddelenie čísel medzerou  
>>v = [1, 2, 3]        % Oddelenie čísel čiarkou
```

Výsledkom je jednoduchý vektor:

```
>>v =  
    1    2    3
```

Na vytvorenie zvislého vektoru je potreba využiť stredník.

```
>>v = [1; 2; 3]
```

Výsledok:

```
v =  
    1  
    2  
    3
```

#### Príklad 2: Počet prvkov vo vektore

```
>>length(v)  
ans =  
    3
```

**Příklad 3: Vytvorenie vektoru s ekvidistantne vzdialenými prvkami**

V preklade to znamená vytvorenie vektoru s určeným začiatkom a koncom, kde prírastok je rovný hodnote 1.

```
>>v = 1:10
v =
    1    2    3    4    5    6    7    8    9   10
```

Ak je potreba určiť krok je potreba pridať ďalší parameter.

```
>>v = 1:1.5:10
v =
    1.0000    2.5000    4.0000    5.5000    7.0000    8.5000   10.0000
```

```
>>v = 10:-2:1
v =
   10    8    6    4    2
```

Parameter prírastku sa vkladá medzi začiatočnú a koncovú hodnotu.

Ďalším spôsobom ako vytvoriť vektor je určiť hranice, z ktorých má MATLAB vytvoriť vektor a následne počet prvkoch, ktoré sú potreba. Využíva sa pri tom príkaz *linspace*.

**Příklad 4: Vytvorenie vektoru s príkazom *linspace***

```
v = linspace (1,3,8)           % Hranice sú 1 až 3 a počet prvkov je 8
v3 =
    1.0000    1.2857    1.5714    1.8571    2.1429    2.4286    2.7143    3.0000
```

**8.2 Matice**

Ako už bolo vysvetlené v teoretickej časti, matica o rozmere  $m \times n$  sa vytvára pomocou hranatých zátvoriek, v ktorých sa uvádzajú jednotlivé riadky oddelené bodkočiarkou. Jednotlivé prvky každého riadku sa oddeľujú medzerou alebo čiarkou. V každom riadku musí byť rovnaký počet prvkov.

**Příklad 1: Zápis medzera, bodkočiarka**

```
>>A = [1 2 3;4 5 6;7 8 9]
A =
    1    2    3
    4    5    6
    7    8    9
```

**Príklad 2: Zápis čiarka, bodkočiarka**

```
>>A = [1,2,3;4,5,6;7,8,9]
```

```
A =  
    1    2    3  
    4    5    6  
    7    8    9
```

**Príklad 3: Zápis čiarka, enter**

```
>>A = [1,2,3
```

```
    4,5,6
```

```
    7,8,9]
```

```
A =  
    1    2    3  
    4    5    6  
    7    8    9
```

V maticiach je taktiež možné po jej vytvorení meniť jednotlivé hodnoty prvkov. Zmena sa vykoná udaním súradníc prvku (riadok, stĺpec) a novej hodnoty daného prvku.

**Príklad 4: Zmena hodnoty prvku matice**

V príklade využijeme maticu A z predchádzajúceho príkladu.

```
>>A(2,2) = 0
```

```
A =  
    1    2    3  
    4    0    6  
    7    8    9
```

Prvok, ktorý sa nachádza v druhom riadku a druhom stĺpci zmenil hodnotu z čísla 5 na číslo 0.

```
>>A(1,1) = 77
```

```
A =  
   77    2    3  
    4    5    6  
    7    8    9
```

Ďalším spôsobom ako nadefinovať prvky matice je tzv. definovanie pozície prvkov v matici. V tomto spôsobe matice sa určuje jednotlivá poloha potrebných prvkov. Aby sa zachovala matica, MATLAB ostatné, nenadefinované prvky nastaví MATLAB na hodnotu 0.

**Príklad 5: Priradenie hodnôt celému riadku a stĺpcu**

```
>>A(1,:) = [5 5 5]
```

*% Priradenie hodnôt celému prvému riadku*

A =

```
5  5  5
4  5  6
7  8  9
```

>>A(:,2) = [0 0 0]

*% Priradenie hodnôt celému druhému stĺpcu*

A =

```
1  0  3
4  0  6
7  0  9
```

#### Príklad 6: Priradenie hodnôt do podmatice

>>A(1:2,[2 3]) = [-5 -8; -7 -9]

A =

```
1  -5  -8
4  -7  -9
7   8   9
```

#### Príklad 7: Vymazanie riadku a stĺpca matice

>>A(1,:) = []

*% Vymazanie celého prvého riadku z matice A*

A =

```
4  5  6
7  8  9
```

>> A(:,2) = []

*% Vymazanie celého druhého stĺpca z matice*

A =

```
1  3
4  6
7  9
```

>> A(end,:) = []

*% Vymazanie celého posledného riadku z matice A*

A =

```
1  2  3
4  5  6
```

#### Príklad 8: Definovanie matice po prvkoch

>>A(1,3) = 3

A =

```
0  0  3
```

>>A(2,1) = 4

A =

```
0  0  3
4  0  0
```

>>A(3,2) = 9

```
A =  
    0    0    3  
    4    0    0  
    0    9    0
```

Týmto spôsobom je možné nadefinovať aj dva prvky naraz.

```
>>M(3,2:3) = [8 9]  
M =  
    0    0    0  
    0    0    0  
    0    8    9
```

### Príklad 9: Vygenerovanie matice

```
>>M=rand(2,3)  
M =  
    0.7922    0.6557    0.8491  
    0.9595    0.0357    0.9340
```

### Príklad 10: Vygenerovanie štvorcovej matice

```
>>M1=rand(4)  
M1 =  
    0.6787    0.6555    0.2769    0.6948  
    0.7577    0.1712    0.0462    0.3171  
    0.7431    0.7060    0.0971    0.9502  
    0.3922    0.0318    0.8235    0.0344
```

U štvorcovej matice stačí zadať len jeden parameter. Príkaz je rovnaký ako

```
>> M1=rand(4,4).
```

### Príklad 11: Vygenerovanie nulovej matice

```
>>M = zeros(2,3)  
M =  
    0    0    0  
    0    0    0
```

### Príklad 12: Vygenerovanie matice zloženej z jednotiek

```
>>M1 = ones(3,2)  
M1 =  
    1    1  
    1    1  
    1    1
```

### Príklad 13: Vygenerovanie jednotkovej matice

```
>>M = eye(3,3)
```

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

#### Príklad 14: Spojenie matice a riadkového vektoru

Matica  $A$  a vektor  $v = [3, 3, 3]$ .

$$>>M = [A; [v]]$$

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 3 & 3 & 3 \end{pmatrix}$$

Ak nie je matica alebo vektor priamo nadefinovaný, je ich možné zadať priamo do zápisu.

$$>>M = [A; [3,3,3]]$$

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 3 & 3 & 3 \end{pmatrix}$$

$$>>M = [[1\ 2\ 3;4\ 5\ 6;7\ 8\ 9];[v]]$$

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 3 & 3 & 3 \end{pmatrix}$$

#### Príklad 15: Spojenie matice a stĺpcového vektoru

$$>>M = [A\ [v]']$$

$$M = \begin{pmatrix} 1 & 2 & 3 & 3 \\ 4 & 5 & 6 & 3 \\ 7 & 8 & 9 & 3 \end{pmatrix}$$

Tak ako v predchádzajúcom príklade je možné maticu alebo vektor nadefinovať priamo do príkazu.

$$>>M = [A\ [3,3,3]']$$

$$>>M = [[1\ 2\ 3;4\ 5\ 6;7\ 8\ 9]\ [v]']$$

Výsledok je u oboch príkazov rovnaký.

$$M = \begin{pmatrix} 1 & 2 & 3 & 3 \\ 4 & 5 & 6 & 3 \\ 7 & 8 & 9 & 3 \end{pmatrix}$$

Tak ako je možné spojovať matice a vektory, je možné spojiť aj dve a viac matíc do jednej.

Nadefinujeme si dve matice. Aby bolo spojenie dobre viditeľné, nadefinujeme si maticu  $A=[3,3;3,3]$  a maticu  $B=[5,5;5,5]$ .

#### Príklad 16: Spojenie dvoch matíc vedľa seba

$$\begin{aligned} >>M=[A \ B] \\ M = \end{aligned} \begin{pmatrix} 3 & 3 & 5 & 5 \\ 3 & 3 & 5 & 5 \end{pmatrix}$$

#### Príklad 17: Spojenie dvoch matíc pod seba

$$M = \begin{pmatrix} 3 & 3 \\ 3 & 3 \\ 5 & 5 \\ 5 & 5 \end{pmatrix}$$

#### Príklad 18: Spojenie viacerých matíc

$$\begin{aligned} >>M=[A \ B;B \ A] \\ M = \end{aligned} \begin{pmatrix} 3 & 3 & 5 & 5 \\ 3 & 3 & 5 & 5 \\ 5 & 5 & 3 & 3 \\ 5 & 5 & 3 & 3 \end{pmatrix}$$

Matice samozrejme musia mať „kompatibilné rozmery“.

### 8.2.1 Výber prvkov a určitej časti matice

Táto podkapitola bude zameraná na základné operácie s maticami, ako je napríklad výber určitého prvku z matice, sčítanie, odčítanie matíc násobenie a delenie matíc.

Pri ukážke jednotlivých operácii bude využitá matica  $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

#### Príklad 19: Výber prvku z matice

$$\begin{aligned} >>prvok = A(1,3) \\ prvok = \\ 3 \end{aligned}$$

*% Výber prvku nachádzajúceho sa na prvom riadku a v treťom stĺpci*

```
>>A(3,3)
ans =
    9
```

*% Výber prvku nachádzajúceho sa na treťom riadku a v treťom stĺpci*

```
>> A(9)
ans =
    9
```

*% Výber prvku pomocou jedného indexu. Je to číslo pozície na ktorej sa nachádza*

```
>>A(3)
ans =
    7
```

*% Výber prvku nachádzajúceho sa na tretej pozícii.*

Hodnota indexu jednotlivých prvkov v matici sa zvyšuje po jednotlivých stĺpcoch. U matici

$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$  má číslo 1 hodnotu indexu 1, číslo 4 má hodnotu indexu 2, číslo 7 má

hodnotu indexu 3, číslo 2 má hodnotu indexu 4 a tak ďalej.

### Príklad 20: Výber riadku a stĺpca

```
>> riadok = A(2,:)
riadok =
    4    5    6
```

*% Výraz „ : “ znamená výber zo všetkých stĺpcov.*

```
>> stlpec = A(:,3)
stlpec =
    3
    6
    9
```

*% Výraz „ : “ znamená výber zo všetkých riadkov.*

Z matice je možné vybrať len určitú časť. Takéto druhý príkazov sa v bežnej praxi využívajú zriedkavo. Uplatnenie je u zložitých výpočtov, pri ktorých sa MATLAB využíva.

### Príklad 21: Výber určitej časti matice

```
>>vyber = A(1:2,2:3)
vyber =
    2    3
    5    6
```

*% Výber prvkov z riadkov 1 až 2 a stĺpcov 2 až 3*

```
>>vyber= A(2:3,[1 2 3])
vyber =
    4    5    6
    7    8    9
```

*% Výber prvkov z riadkov 2 a 3 a stĺpcov 1 ,2 a 3*

```
>>A(2,1:2)
ans =
    4    5
```

*% Výber prvkov z riadku 2 a stĺpcov 1 až 2*



**Příklad 22: Výber celého riadku matice**

```
>>A(2,:)
ans =
    4    5    6
```

```
>>A(1,:)
ans =
    1    2    3
```

**Příklad 23: Výber celého stĺpca matice**

```
>>A(:,3)
ans =
    3
    6
    9
```

**Příklad 24: Pretransformovanie matice na stĺpcový vektor**

```
>>A(:)
ans =
    1
    4
    7
    2
    5
    8
    3
    6
    9
```

**8.2.2 Funkcie pre manipuláciu s maticami**

Táto podkapitola bude zameraná na funkcie pre manipuláciu s maticami.

**Příklad 25: Výber hlavnej diagonály matice**

```
>>D = diag(A)
D =
    1
    5
    9
```

**Příklad 26: Výber subdiagonály pod hlavnou diagonálou matice**

```
>>D = diag(A,-1)
D =
    4
    8
```

**Příklad 27: Výber subdiagonály nad hlavnou diagonálou matice**

```
>>D = diag(A,1)
```

```
D =
    2
    6
```

```
>>D = diag(A,2)
```

```
D =
    3
```

**Příklad 28: Vytvorenie diagonály**

Pre ukážku tohto príkazu je potreba definovať vektor, ktorý sa využije ako diagonála v matici

```
>>diagonala= [1 2 3 4];
```

*% Vektor, ktorý je využitý ako diagonála*

```
>>D = diag(diagonala)
```

```
D =
    1    0    0    0
    0    2    0    0
    0    0    3    0
    0    0    0    4
```

**Příklad 29: Záměna stĺpcov a riadkov**

```
>>B = fliplr(A)
```

*% Záměna stĺpcov*

```
B =
    3    2    1
    6    5    4
    9    8    7
```

```
>>B = flipud(A)
```

*% Záměna riadkov*

```
B =
    7    8    9
    4    5    6
    1    2    3
```

**Příklad 30: Rozmer matice**

```
>>size(A)
```

*% Rozmer matice A*

```
ans =
    3    3
```

*% Rozmer je teda 3x3*

```
>>size(D)
```

```
ans =
    4    4
```

**Príklad 31: Rotácia matice**

```
>>rotacia = rot90(A)                                % Rotácia matice o 90°
rotacia =
     3     6     9
     2     5     8
     1     4     7
```

V prípade použitia toho príkazu ešte raz na posunutú maticu, novovzniknutá matica bude rotovať o ďalších 90°.

```
>>rotacia2 = rot90(rotacia)
rotacia2 =
     9     8     7
     6     5     4
     3     2     1
```

Ak je potreba hneď na začiatku posunúť maticu o viac ako 90°, je možné využiť príkaz *rot90 (A,k)*, kde A je matica a k je počet rotácií o 90°. tento príkaz je vhodný najmä pri zložitejších funkciách, kde nám pomáha pri sprehl'adnení kódu.

Rotácia o 180° teda bude vyzeráť nasledovne:

```
>>rotacia = rot90 (A,2)
rotacia =
     9     8     7
     6     5     4
     3     2     1
```

**Príklad 32: Vytvorenie dolnej trojuholníkovej matice**

```
>>B = tril(A)
B =
     1     0     0
     4     5     0
     7     8     9
```

MATLAB dolnú trojuholníkovú maticu vytvorí tak, že prvky ktoré sa nachádzajú nad hlavnou diagonálou nahradí hodnotou 0. Ak je potreba odstrániť aj hlavnú diagonálu musíme zátvorku doplniť o parameter -1 (A,-1). Týmto parametrom nahradíme prvky hlavnej diagonály hodnotou 0. Tento parameter, v prípade väčších matíc je možné zmenšovať, čím sa dosiahne menšia a menšia dolná trojuholníková matica. Prvky jednotlivých subdiagonál sa pri znižovaní tohto parametru postupne nastavujú na hodnotu 0.

```
>>B = tril(A,-1)
```

```

B =
    0    0    0
    4    0    0
    7    8    0

>>B = tril(A,-2)
B =
    0    0    0
    0    0    0
    7    0    0

```

### Príklad 33: Vytvorenie hornej trojuholníkovej matice

```

>>B = triu(A)
B =
    1    2    3
    0    5    6
    0    0    9

```

Postup je rovnaký ako v predchádzajúcom príklade. V tomto prípade, ale MATLAB nenastaví hodnotu 0 prvkom, ktoré sú nad diagonálou, ale pod diagonálou. Ak je potreba odstrániť aj hlavnú diagonálu, musíme zátvorku doplniť o parameter 1 (A,1). U väčších matic je možné tento parameter zväčšovať, čím sa dosiahne menšia a menšia horná trojuholníková matica. Prvky jednotlivých subdiagonál sa pri zvyšovaní tohto parametru postupne nastavujú na hodnotu 0.

```

>>B = triu(A,1)
B =
    0    2    3
    0    0    6
    0    0    0

```

```

>>B = triu(A,2)
B =
    0    0    3
    0    0    0
    0    0    0

```

### 8.2.3 Maticové operácie

Táto podkapitola bude zameraná na maticové operácie.

```
A = [1 4; 5 9]
```

```
B = [6 -1; -4 3]
```

**Příklad 34: Súčet matíc**

```
>>A + B
ans =
    7    3
    1   12
```

```
>>15+A
```

```
ans =
   16   19
   20   24
```

**Příklad 35: Rozdiel matíc**

```
>>A-B
ans =
   -5    5
    9    6
```

```
>>B-8
```

```
ans =
   -2   -9
  -12   -5
```

**Příklad 36: Násobenie matíc**

```
>>A*B
ans =
  -10   11
   -6   22
```

```
>>C = [5;3];
```

*% Vytvorenie stĺpcovej matice*

```
>>A*C
ans =
   17
   52
```

```
>>B*5
```

*% Vynásobenie prvkov matice B číslom 5*

```
ans =
   30   -5
  -20   15
```

**Příklad 37: Násobenie matíc po prvkoch**

V násobení po prvkoch dochádza k vzájomnému vynásobeniu prvkov na rovnakých miestach v maticiach vstupujúcich do násobenia. V MATLABe sa tento úkon realizuje prida-

ním bodky pre príslušný operátor. (Operátor bodka vždy mení význam nasledujúceho operátora tak, aby bol aplikovaný po prvkoch).

```
>>A.*B
ans =
     6    -4
    -20    27
```

### Príklad 38: Delenie zľava = $A*B^{-1}$

Ide o násobenie inverznou maticou. Existujú dva druhy delenia matíc. Ľavé a pravé.

```
>>A/B
ans =
    1.3571    1.7857
    3.6429    4.2143
```

```
>>A*inv(B)
ans =
    1.3571    1.7857
    3.6429    4.2143
```

```
>> B/5
ans =
    1.2000   -0.2000
   -0.8000    0.6000
```

### Príklad 39: Delenie sprava = $A^{-1}*B$

```
>>A\B
ans =
   -6.3636    1.9091
    3.0909   -0.7273
```

```
>>inv(A)* B
ans =
   -6.3636    1.9091
    3.0909   -0.7273
```

### Príklad 40: Delenie matíc po prvkoch

```
>>A./B
ans =
    0.1667   -4.0000
   -1.2500    3.0000
```

```
>>A.\B
ans =
    6.0000   -0.2500
   -0.8000    0.3333
```

**Príklad 41: Umocnenie matice**

```
>>A^2                                % Je možné využiť aj násobenie matic A*A...
ans =
    21    40
    50   101
```

**Príklad 42: Umocnenie matice po prvkoch**

```
>>A.^2
ans =
     1    16
    25    81
```

**Príklad 43: Transpozícia matice**

```
>>A'
ans =
     1     5
     4     9
```

V skutočnosti je  $A'$  matica (komplexne) konjugovaná k matici  $A$ , to znamená, že ak  $A$  obsahuje komplexné čísla s nenulovou imaginárnou časťou, tak po transpozícii sú u imaginárnych častí opačné znamienka.

Pravá transponovaná matica vyzerá takto:

```
>>X=[1 2 3; 4 5 6];
>>Y = X+i*(rand(2,3)-0.5)
Y =
 1.0000 + 0.2094i 2.0000 - 0.2240i 3.0000 + 0.1551i
 4.0000 + 0.2547i 5.0000 + 0.1797i 6.0000 - 0.3374i
>>Y'
ans =
 1.0000 - 0.2094i 4.0000 - 0.2547i
 2.0000 + 0.2240i 5.0000 - 0.1797i
 3.0000 - 0.1551i 6.0000 + 0.3374i
```

Ak aj v prípade komplexných čísel potrebujeme iba transponovanú maticu bez zmeny znamienok, tak sa pred apostrof vloží bodka.

```
>>Y.'
ans =
 1.0000 + 0.2094i 4.0000 + 0.2547i
 2.0000 - 0.2240i 5.0000 + 0.1797i
 3.0000 + 0.1551i 6.0000 - 0.3374i
```

**Příklad 44: Vlastné čísla matice**

$A = [1 \ 4; 5 \ 9]$

```
>> eig(A)
```

```
ans =
```

```
    -1
```

```
    11
```

```
>> X = [5, -7, 12; 7, -8, 6; 12, -8, 4 - 21];
```

```
>> eig(X)
```

```
ans =
```

```
 -21.5358
```

```
  2.6427
```

```
 -1.1069
```

**Příklad 45: Hodnost' matice**

$A = [1 \ 4; 5 \ 9];$

```
>> rank(A)
```

```
ans =
```

```
    2
```

```
>> X = [5, -7, 12; 7, -8, 6; 12, -8, 4 - 21];
```

```
>> rank(X)
```

```
ans =
```

```
    3
```

**Příklad 46: Determinant matice**

$A = [1 \ 4; 5 \ 9]$

```
>> det(A)
```

```
ans =
```

```
   -11
```

```
>> X = [5, -7, 12; 7, -8, 6; 12, -8, 4 - 21];
```

```
>> det(X)
```

```
ans =
```

```
  63.0000
```

**Příklad 47: Logické operácie**

$A = [1 \ 4; 5 \ 9]$

```
>> B = A > 5
```

```
B =
```

```
    0    0
```



0 1

MATLAB pri logických operáciách využíva log0 a log1. V prípade, že hodnota v matici splňuje zadanú podmienku, označí jej pozíciu hodnotou 1, v opačnom prípade pozíciu v matici označí hodnotou 0.

```
>>X= [5,-7,12; 7,-8,6;12,-8,4-21];
```

```
>>C= X>3
```

```
C =  
    1    0    1  
    1    0    1  
    1    0    0
```

```
>>X= [5,-7,12; 7,-8,6;12,-8,4-21];
```

```
>>D= X<-2
```

```
D =  
    0    1    0  
    0    1    0  
    0    1    1
```

```
>>X= [5,-7,12; 7,-8,6;12,-8,4-21];
```

```
>>E= (X>=6)
```

*% Zátvorky ( ) nie sú povinné, je to z dôvodu prehľadnosti*

```
E =  
    0    0    1  
    1    0    1  
    1    0    0
```

Ak je potrebné vedieť ich počet, tak len spočítame jednotky v matici.

```
>>sum(sum(E))
```

```
ans =  
    4
```

```
>>sum(sum(B))
```

```
ans =  
    1
```

#### 8.2.4 Riešenie sústav lineárnych algebraických rovníc

Táto podkapitola bude zameraná na riešenie sústav lineárnych algebraických rovníc.

Pri riešení sústavy lineárnych algebraických rovníc je možné využiť operáciu delenia matíc zľava.

**Príklad 48: Riešenie sústavy lineárnych algebraických rovníc**

$$5x_1 + 7x_2 + x_3 = 3$$

$$4x_1 + 6x_2 + 3x_3 = 12$$

$$6x_1 + 7x_2 + 5x_3 = 4$$

```
>>A=[5,7,1;4,6,3;6,7,5]
```

```
A =
```

```
5    7    1
4    6    3
6    7    5
```

```
>>b=[3;12;4]
```

```
b =
```

```
3
12
4
```

Pre výpočet je možné využiť dva spôsoby. Ľavo stranové delenie alebo operáciu inverznej matice.

```
>>x = inv(A)*b
```

```
x =
```

```
-10.8261
 7.7391
 2.9565
```

```
>>x=A\b
```

```
x =
```

```
-10.8261
 7.7391
 2.9565
```

Kontrola správnosti výsledku

```
>>A*x-b
```

```
ans =
```

```
1.0e-14 *
-0.3109
-0.8882
-0.1776
```

Ako je vidieť u výsledkov, oba prípady sú vhodné na výpočet koreňov lineárnych algebraických rovníc. Niektoré učebnice odporúčajú pri výpočte využívať ľavo stranové delenie namiesto operácie inverzie matice. Uvádzajú, že výpočet pomocou inverzie matice je u zložitejších výrazov menej presné a môže viesť k nestabilite až k zrúteniu programu. (Perůtka, 2005, 55s.)

### 8.3 Neriešené príklady

1. Vytvorte vektor  $V1 = (1, 5, 9, 7, 3)$  a  $V2 = (1, 4, 9, 6, 2)$ .
2. Vytvorte vektor, ktorý bude mať 8 prvkov, ktorého hodnoty budú mať pravidelný odstup a budú v rozmedzí od 3 do 7.
3. Vytvorte maticu  $M1 = \begin{pmatrix} 6 & 7 & 5 & 6 \\ -8 & 1 & -3 & 9 \\ 4 & -7 & 1 & 4 \\ 3 & 4 & -9 & -5 \end{pmatrix}$ .
4. Vygenerujte štvorcovú maticu  $M2 = 4 \times 4$ .
5. Sčítajte maticu  $M1$  a  $M2$ , výslednú maticu podel'te hodnotou 4 a následne umocnite hodnotou 3.
6. Maticu  $M1$  vynásobte po prvkoch s maticou  $M3 = \begin{pmatrix} 7 & 3 & 13 & 3 \\ 5 & -9 & 4 & 7 \\ -3 & 7 & 9 & 1 \\ 1 & 14 & -3 & 7 \end{pmatrix}$ .
7. Maticu  $M3$  podel'te maticou  $M1$  pomocou operácie delenie zľava.
8. Vypíšte 2. a 3. riadok matice  $M1$ . Vypíšte 1 a 4 stĺpec matice  $M1$ .
9. V matici  $M3$  zmeňte hodnoty prvého riadku na číslo 1 a hodnoty posledného stĺpca na 9.
10. Vymažte 2. a 3. riadok matice  $M3$ .
11. K matici  $M1$  pripojte vektor  $v1$ , následne zmeňte vektor  $v2$  na stĺpcový a pripojte ho taktiež k matici  $M1$ .
12. Vygenerujte maticu  $M4$  zloženú z jednotiek, ktorá bude mať rozmer  $4 \times 4$  a pripojte ho vedľa matice  $M1$  a následne aj pod maticu  $M1$ .
13. Vyberte prvky z 3. a 4. riadku a 1. až 3. stĺpca matice  $M3$ .
14. Vyberte prvky z 2. riadku a 1., 3. a 4. stĺpca  $M1$ .
15. Vyberte prvky z 2. a 4. riadku a zo stĺpcov 3 a 4.
16. Zobrazte hlavnú diagonálu z matice  $M1$ .
17. Vytvorte hlavnú diagonálu z vektoru  $V2$ .
18. Zobrazte subdiagonálu nad hlavnou diagonálou z matice  $M3$ .
19. Zobrazte trojuholníkové matice z matice  $M3$  pod a nad hlavnou diagonálou.
20. Pretransformujte maticu  $M1$  na stĺpcový vektor.
21. Vypočítajte vlastné čísla a hodnotu matice  $M3$ .
22. Vypočítajte determinant matice  $M1$ .
23. Vytvorte rotáciu o  $180^\circ$  u matice  $M3$ .

## 9 KOMPLEXNÉ ČÍSLA

V tejto kapitole budú ďalej rozobrané komplexné čísla. Komplexné čísla pozostávajú z dvoch častí, a to reálnej časti a imaginárnej časti.

### Príklad 1: Vytvorenie komplexného čísla

```
>>x = 5+5i
x =
  5.0000 + 5.0000i

>>y = complex(5,5)
y =
  5.0000 + 5.0000i
```

Pomocou príkazu whos je možné zistiť či sa jedná o komplexné číslo.

```
>>whos x
Name      Size      Bytes Class Attributes
x         1x1         16 double complex
```

### Príklad 2: Zadávanie komplexných čísel

```
>>x= 12+9*i
x =
  12.0000 + 9.0000i

>>y=9+i*7
y =
  9.0000 + 7.0000i
```

### Príklad 3: Jednoduché aritmetické operácie s komplexnými číslami

```
>>plus = x+y
plus =
  21.0000 + 16.0000i

>>plus2 = x+10
plus2 =
  22.0000 + 9.0000i

>>minus = x-y
minus =
  3.0000 + 2.0000i

>>minus2 = y-3.5
minus2 =
  5.5000 + 7.0000i

>>sucin = x*y
sucin =
  4.5000e+01 + 1.6500e+02i
```

```
>>podiel = x/y
podiel =
1.3154 - 0.0231i

>>umocnenie= x^2
umocnenie =
6.3000e+01 + 2.1600e+02i

>>odmocnina= sqrt(x)
odmocnina =
3.6742 + 1.2247i
```

#### Príklad 4: Výber len reálnej alebo komplexnej časti

```
>>xreal = real(x)
xreal =
12

>>yimag = imag(y)
yimag =
7
```

Treba si dávať pozor nato, že časť získaná príkazom *imag* už nie je komplexné číslo.

```
>>whos yimag
Name      Size      Bytes Class Attributes
yimag     1x1           8 double
```

Na vyhodnotenie či je dané číslo imaginárne alebo len reálne je možné využiť funkciu *isreal*, ktorá vracia logickú premennú 1 alebo 0. Komplexné číslo s nulovou imaginárnou časťou sa taktiež považuje za reálne. (Blaho, 2007)

```
>>x= 12+9*i;
>>isreal(x)
ans =
0

>>isreal(5+0i)
ans =
1

>>isreal(yimag)
ans =
1
```

#### Príklad 5: Absolútna hodnota komplexného čísla

```
>>z=abs(x)
z =
15
```

Príkaz pre absolútnu hodnotu je skrátenejší verzia príkazu  $\text{sqrt}(\text{real}(X).^2 + \text{imag}(X).^2)$

```
>>z = sqrt(real(x).^2 + imag(x).^2)
```

```
z =  
    15
```

```
>>z=abs (y)
```

```
z =  
    11.4018
```

```
>> z = sqrt(real(y).^2 + imag(y).^2)
```

```
z =  
    11.4018
```

Príkaz neslúži len pre komplexné čísla, ale aj pre čísla ktoré sú reálne.

```
>>z=abs (-25)
```

```
z =  
    25
```

## 9.1 Neriešené príklady

1. Vytvorte komplexné čísla  $k_1 = 7+12i$  a  $k_2 = 6-5i$
2. Vytvorte súčet komplexných čísel  $k_1$  a  $k_2$
3. Vytvorte súčin komplexných čísel  $k_1$  a  $k_2$
4. Vytvorte druhú odmocninu z komplexného čísla  $k_1$
5. Vytvorte absolútnu hodnotu z komplexného čísla  $k_2$
6. Vypíšte imaginárnu časť z komplexného čísla  $k_1$  a reálnu časť z komplexného čísla  $k_2$

## 10 PROGRAMOVANIE PODMIENOK A CYKLOV

V tejto kapitole bude vysvetlené programovanie podmienok a cyklov.

Pre programovanie podmienok a cyklov je potreba využiť M-súbory, nakoľko podmienky a cykly obsahujú viac príkazov.

### 10.1 Podmienka IF

#### Príklad 1: Podmienka IF

```
a=12;
b=-5;
v=a+b;
if v > 7
    disp('Sucet je vacsi nez 7');
elseif v==7,
    disp('Sucet je 7');
else
    disp('Sucet je mensi nez 7');
end
```

V príkazovom riadku sa nám objavia 2 riadky. Prvý riadok je názov M-súboru, ktorý sa spustí a druhý riadok je výstup z M-súboru

```
>>cykly
Sucet je 7
```

#### Príklad 2: Podmienka IF

```
prem= 'Zadajte cislo a ';
a=input(prem);                                % Načítanie zadaného čísla do premennej a
if a>10
    disp('Cislo a je vacsie ako 10 ')
elseif a==10
    disp('Cislo a je 10 ')
else
    disp('Cislo a je mensie ako 10 ')
end
```

Po spustení M-súboru sa v príkazovom riadku zobrazí názov spusteného M-súboru, (>> cykly), a požiadavka na zadanie čísla (Zadajte cislo a). Od užívateľa potom závisí, aké číslo zadá a teda ktorá z možností podmienky sa mu zobrazí.

### 10.2 Vetvenie pomocou switch

```
prem= 'Zadajte cislo a ';                    % Požiadavok na zadanie čísla a
a=input(prem);                                % Načítanie zadanej hodnoty do premennej a
```

```

prem1 = 'Zadajte cislo b ';           % Požadavok na zadanie čísla a
b=input(prem1);                       % Načítanie zadanej hodnoty do premennej a
disp('-----');                     % Pomocne čiary na oddelenie menu
disp('Vyberte jednu z možností:');
disp('1. Sčítanie');
disp('2. Rozdiel');                   % Výpis Menu
disp('3. Násobenie');
disp('4. Delenie');
prem3 = 'Vasa volba: ';               % Požadavok na zadanie čísla z menu
r=input(prem3);                       % Načítanie zadanej hodnoty menu do premennej r

disp('-----');
switch r                               % Zistenie zadanej hodnoty a výber príslušného
case 1
    case 1
        v=a+b;
        fprintf('Sucet je: %d \n',v); % Výpis „Sucet je“ a následne výsledku v
    case 2
        v=a-b;
        fprintf('Rozdiel je: %d \n',v); % Výpis „Rozdiel je“ a následne výsledku v
    case 3
        v=a*b;
        fprintf('Sucin je: %d \n',v); % Výpis „Sucin je“ a následne výsledku v
    case 4
        v=a/b;
        fprintf('Delenie je: %d \n',v); % Výpis „Delenie je“ a následne výsledku v
end

```

V príklade sa objavil nový znak „%d“. Znamená to výpis čísla typu double. Ďalším novým znakom je „\n“. Funkcia tohto znaku je posunutie o jeden riadok nižšie. V tomto príklade to znamená, že po vypísaní výsledku je potreba odriadkovať.

Spustenie M-súboru:

```

>>cykly
Zadajte cislo a:                       % Zadanie hodnôt A a B
Zadajte cislo b:
-----
Vyberte jednu z možností:             % Hlavné menu
1. Sčítanie
2. Rozdiel
3. Násobenie
4. Delenie
Vasa volba:                           % Zadanie výberu z menu
-----

```

Po zadání voľby z hlavného menu sa zobrazí jeden zo štyroch výsledkov.



*Sučet je:*

*Rozdiel je:*

*% Vypísanie jedného z textov podľa voľby  
v menu a následný výsledok*

*Sucin je:*

*Delenie je:*

Každopádne program nie je celkom správne, pretože je rozdiel ak odčítame B od A a naopak. Je nutné do možnosti 2 a 4 pridať ďalší switch alebo if.

Doplnenie pomocou switch:

*case 2*

<i>disp('1. A-B');</i>	<i>% Vypísanie podmenu</i>
<i>disp('2. B-A');</i>	
<i>prem4= 'Vasa volba: ';</i>	<i>% Výber voľby z menu</i>
<i>t=input(prem4);</i>	<i>% Zapísanie voľby do premennej</i>
<i>switch t</i>	<i>% Zistenie zadanej hodnoty a výber príslušného case</i>
<i>case 1</i>	<i>% Case pre rozdiel A-B</i>
<i>    v=a-b;</i>	
<i>fprintf('Rozdiel je: %d \n',v);</i>	<i>% Výpis „Rozdiel je“ a následne výsledku v</i>
<i>case 2</i>	<i>% Case pre rozdiel B-A</i>
<i>    v=b-a;</i>	
<i>fprintf('Rozdiel je: %d \n',v);</i>	<i>% Výpis „Rozdiel je“ a následne výsledku v</i>
<i>end</i>	

*case 4*

```

disp('1. A/B');
disp('2. B/A');
prem4= 'Vasa volba: ';
t=input(prem4);
switch t
case 1
    v=a/b;
    fprintf('Podiel je: %d \n',v);
case 2
    v=b/a;
    fprintf('Podiel je: %d \n',v);
end

```

Výsledkom je potom presnejší program, ktorý obsahuje presnejší rozdiel a podiel čísel A a B. Tento druh vetvenia programu nie je tak často využívaný. Častejšie sa pri tomto jednoduchom menu využije podmienka if.

Doplnenie pomocou podmienky if:

*case 2*

```
disp('1. A-B');
disp('2. B-A');
prem4= 'Vasa volba: ';
x=input(prem4);
if x==1
    v=a-b;
fprintf('Rozdiel je: %d \n',v);
else
    v=b-a;
fprintf('Rozdiel je: %d \n',v);
end
```

*case 4*

```
disp('1. A/B');
disp('2. B/A');
prem4= 'Vasa volba: ';
x=input(prem4);
if x==1
    v=a/b;
fprintf('Podiel je: %d \n',v);
else
    v=b/a;
fprintf('Podiel je: %d \n',v);
end
```

Celý tento program je taktiež možné napísať len pomocou podmienky if. Stačí len switch zameniť za podmienku if a jednotlivé case za elseif, ktoré by testovali rovnosť z jednotlivými hodnotami, ktoré boli uvedené v hlavnom menu. Je na užívateľovi, ktorý spôsob mu viac vyhovuje a taktiež ktorý spôsob je v danom programe prehľadnejší.

Výsledkom doplnenia podmienky if alebo switch. Zadané čísla sú A=6 a B=2.

*Vasa volba: 2*

-----  
1. A-B

2. B-A

Vasa volba: 1

Rozdiel je: 4

Vasa volba: 2

-----  
1. A-B

2. B-A

Vasa volba: 2

Rozdiel je: -4

Vasa volba: 4

-----

1. A/B

2. B/A

Vasa volba: 1

Podiel je: 3

Vasa volba: 4

-----

1. A/B

2. B/A

Vasa volba: 2

Podiel je: 3.333333e-01

### Príklad 3: Funkcia vetvenia

```
x=[1 3 7 6 5];           % Vytvorenie vektoru
n = length(x);           % Zistenie veľkosti vektoru
switch n
    case 0                 % Ak je vektor nulový
        error('Vektor je prázdny');
    case {1,2}             % Ak má vektor 1 alebo 2 prvky
        disp('Vektor má 1 alebo 2 prvky');
    case {3,4}             % Ak má vektor 3 alebo 4 prvky
        disp('Vektor má 3 alebo 4 prvky');
    otherwise
        disp('Vektor má viac ako 4 prvky'); % Ak má vektor 4 a viac prvkov
end
```

Ďalší príklad vetvenia pomocou príkazu switch sa nachádza v prílohe P I.

### 10.3 Cyklus while

Cyklus while opakuje príkazy pokým je splnená podmienka. Na začiatku sa testuje platnosť riadiacej podmienky. Pokiaľ podmienka platí, t.j. výraz nemá nulovú hodnotu, vykonajú sa všetky príkazy tela cyklu. V ďalšom kroku sa opäť testuje podmienka a pokiaľ platí, vykonajú sa opäť všetky príkazy, ktoré telo cyklu obsahuje. Opakovanie prebieha dovtedy, kým sa pri testovaní podmienky nezistí, že podmienka už neplatí (t.j. hodnota výrazu je nulová). (Majerová, 2013)

```
i = 1;                   % Nastavenie počiatočnej hodnoty
while i < 10              % Kontrola platnosti podmienky (ak platí,
                           pokračuje ďalej, ak neplatí, tak sa program
                           ukončí)

    fprintf('%d ',i);
    i = i+1;
end
```

**Príklad 4: Nekonečný cyklus**

```
while 1                                % Podmienka je vždy pravdivá, má stále
                                        hodnotu 1
    disp('opakujem...')
end
```

**Príklad 5: Výpočet faktoriálu**

```
preml= 'Zadajte cislo n ';
n=input(preml);                        % Načítanie zadanej hodnoty do premennej n
if n<0
    error('faktorial neexistuje')      % Chybové hlásenie ak je n menšie ako 0
end
f= 1;                                  % Nastavenie premennej f na hodnotu 1
while n>1                              % Ak je n väčšie ako 1 pokračuj ďalej inak
                                        ukonči cyklus
    f= f*n;                             % Výpočet faktoriálu
    n= n-1;
end
fprintf('faktoriál je: %d \n',f)        % Výpis hodnoty faktoriálu
```

**10.4 Cyklus for**

Je to cyklus s vopred známym počtom opakovaní. Označuje sa tiež ako iteračný cyklus. Využíva sa teda v prípade, kde je vopred známy počet opakovaní príkazov v tele tohto cyklu. Počet opakovaní je daný vektorom, z ktorého si v každej iterácii (v každom prechode cyklom) zoberieme jednu hodnotu. Táto hodnota je uložená v tzv. riadiacej premennej cyklu. (Majerova, 2013)

```
for i = 1:10
    fprintf('%d ',i);
end
```

**Príklad 6: Naplnenie matice súčinom jej súradníc**

```
preml= 'Zadajte pocet stlpcov: ';
n=input(preml);
prem2= 'Zadajte pocet riadkov: ';
m=input(prem2);
for i=1:m                              % Vytváranie stlpcov
    for j=1:n                          % Vytváranie riadkov
        A=0;                          % Nulovanie premennej A
        A(i,j) = i*j                  % Súčin súradníc
    end
end
```

Po spustení a vložení počtu riadkov a stĺpcov MATLAB postupne v príkazovom riadku vytvára maticu. Koncovým výsledkom potom je:

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 6 \end{pmatrix}$$

### 10.5 Neriešené príklady

1. Vytvorte program s podmienkou *if*, ktorý bude po zadaní čísla zisťovať, či zadané číslo je kladné alebo záporné. V prípade nuly vypíšte vetu: Číslo je rovné NULE.
2. Vytvorte program pomocou vetvenia *switch*, ktoré bude slúžiť ako simulácia reštaurácie. Bude obsahovať základné menu zložené aspoň s piatich vecí, následne výzvu na výber jednej z možností. Po výbere sa zobrazí názov vybraného jedla a cena za toto jedlo.
3. Vytvorte program pomocou cyklu *while*, ktorý bude slúžiť ako odpočítavanie od 20. Pred ukončenie cyklu, vložte príkaz *pause(0.5);*, ktorý slúži ako pauza medzi každým jedným priebehom v cykle.
4. Vytvorte počítadlo pomocou príkazu *for*, ktoré bude počítat' od 1 do 10 a krokom 0,2. Pred ukončenie cyklu, vložte príkaz *pause(0.5);*, ktorý slúži ako pauza medzi každým jedným priebehom v cykle.

## 11 REŤAZCE

V tejto kapitole bude vysvetlené ako sa vytvárajú reťazce.

Znak je v MATLABE reprezentovaný ako celočíselná hodnota konvertovaná na jej Unicode znakový ekvivalent. Na deklarovanie znaku využijeme apostrofy, medzi ktoré napíšeme požadovaný znak. Základné znaky sú reprezentované číslami 32 – 127. Čísla menšie ako 32 sú netlačiteľné riadiace znaky. Ďalšie čísla reprezentujú znaky rôznych fontov na počítači. Číselnú hodnotu znaku môžeme potom získať napríklad príkazom `uint8`.

Textové premenné, sa podobne ako znakové, deklarujú pomocou apostrofov. Text je uložený ako pole znakov.

```
>>znak='P';  
>>cislo=uint8(znak);  
cislo=  
      80
```

### Príklad 1: Číselné hodnoty jednotlivých znakov

```
>>retazec='MATLAB';  
>>hodnoty=uint8(retazec)  
hodnoty =  
      77  65  84  76  65  66
```

Čísla môžeme konvertovať na znaky alebo textové reťazce pomocou príkazu `char`.

### Príklad 2: Prevod číselných hodnôt na reťazec

```
>>priklad=char([80 114 101 118 111 100 32 99 105 115 101 108 32 110 97  
32 114 101 116 97 122 101 99])  
priklad =  
Prevod cisel na retazec
```

Spojenie reťazcov do jedného riadku môžeme urobiť pomocou hranatých zátvoriek a znaky, alebo reťazce oddeliť čiarkou, alebo medzerou. Spojenie v riadku môžeme urobiť aj pomocou príkazu `strcat`.

### Príklad 3: Spojenie reťazcov

```
>>retazec1='Dobry';  
>>retazec2='den';
```

Pomocou hranatých zátvoriek:

```
>> spojenie=[retazec1, ' ', retazec2]  
spojenie =
```

*Dobry den*

```
>>spojenie1=strcat(retazec1,'_',retazec2)
spojenie1 =
Dobry_den
```

Spojenie reťazcov v stĺpci sa robí podobne ako v predchádzajúcom príkaze, ale namiesto čiarky použijeme bodkočiarku. Pri vytváraní polí znakov treba dávať pozor nato, aby mali jednotlivé riadky rovnaký počet znakov. Konce krátkych znakov môžete dopĺňať medzerami.

#### Príklad 4: Spojenie reťazcov do stĺpca

```
spojenie3=[retazec1; retazec2];           % Tento príkaz vyhlási chybu! Nie je rovnaký
>>spojenie3=['Dobry';'den ']              počet znakov
spojenie3 =
Dobry
den                                           % Za reťazcom sú ešte 2 medzery

>> spojenie4=char('Dobry','den')
spojenie4 =
Dobry
den                                           % Za reťazcom sú ešte 2 medzery
```

#### Príklad 5: Nahradenie slov v reťazci

```
>>retazec5='Auto je modrej farby';
>>nahrada=regexprep(retazec5,'Auto','Motorka')
nahrada =
Motorka je modrej farby
```

V Tab. 8 sa nachádza súhrn príkazov na prevod medzi číslami a textovými reťazcami a naopak.

Tab. 8 Príkazy pre prevod medzi číselnými a znakovými typmi (Blaho, 2009)

Funkcia	Popis
<i>char</i>	konvertuje na znakové pole
<i>double</i>	konvertuje pole znakov na číselné hodnoty
<i>int2str</i>	konvertuje celé číslo na znakové pole
<i>mat2str</i>	konvertuje maticu na znakové polia
<i>num2str</i>	konvertuje číslo na znakové pole
<i>str2num</i>	konvertuje znakové pole na číselnú hodnotu
<i>str2double</i>	konvertuje znakové pole na číslo s pohyblivou čiarkou

### 11.1 Neriešené príklady

1. Vypíšte číselné hodnoty z ASCII tabuľky pre textový reťazec r1= Opakovanie
2. Vytvorte textový reťazec r2= Spojenie a r3= retazcov
3. Spojte reťazce r2 a r3 do r4 vedľa seba a oddel'ite ich medzerou
4. Spojte reťazce r2 a r3 do r5 pod seba
5. V reťazci r4 nahrad'ite slovo „retazcov“ za slovo „narodov“



## 12 GRAFY

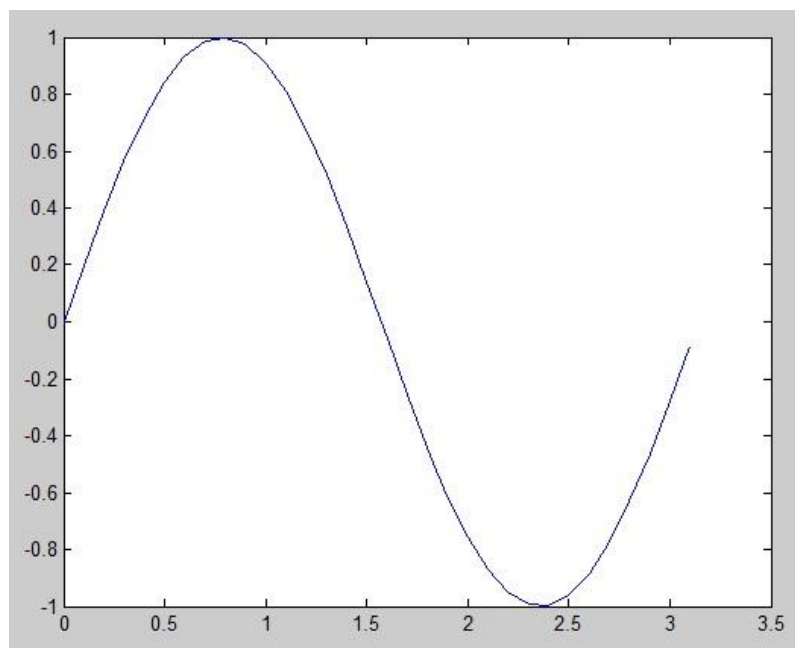
V tejto kapitole bude vysvetlené programovanie grafov, ktoré budú následne aj zobrazené prostredníctvom obrázkov.

### 12.1 2D Grafy

MATLAB disponuje veľkým množstvom funkcií pre tvorbu grafov. Medzi najčastejšie používané patrí funkcia plot.

#### Príklad 1: Graf funkcie $\sin(2*t)$

```
>>t = 0:0.1:pi;  
>>graf = sin(2*t);  
>>plot(t,graf)
```



Obr. 14 Graf funkcie  $\sin(2*t)$

(Vlastné spracovanie)

#### Príklad 2: Výpočet funkcie $1/(2 + 2x^2)$

```
>>x = -2:0.1:2;  
>>y = 1/(2+2*x.^2);  
>>plot(x,y)
```

Graf funkcie sa nachádza v prílohe P II.

U grafov je možné nastaviť farbu čiary, vyznačenie bodov a taktiež vzor čiary. Tieto parametre sa zapisujú nasledovným spôsobom: plot(x,y,'farba\_typ\_znak').

Grafy následujících funkcí sa nachádzajú v prílohe P II.

```
>>plot(x,y,'r')
```

```
>>plot(x,y,'b:')
```

```
>>plot(x,y,'mx')
```

V Tab. 9 a Tab. 10 sú zobrazené grafické funkcie, ktoré sa používajú pri zmene farby, typu, znaku alebo čiary.

Tab. 9 Grafické funkcie zobrazenia

(UIAM FCHPT STU, 2015)

Symbol	Farba (RGB)	Symbol	Typ čiary
c	cyan (0 1 1)	-	plná čiara (auto)
m	magenta (1 0 1)	--	čiarkovaná
y	žltá (1 1 0)	-.	bodkočiarkovaná
r	červená (1 0 0)	:	bodková
g	zelená (0 1 0)	none	žiadna
b	modrá (0 0 1)		
w	biela (1 1 1)		
k	čierna (0 0 0)		

Tab. 10 Grafické funkcie zobrazenia

(UIAM FCHPT STU, 2015)

Znak	Opis znaku
+	plus
o	kruh
*	hviezdička
.	bodka
x	križik
square	štvrtok
diamond	kosoštvorec
V	dole orientovaný trojuholník
<	vľavo orientovaný trojuholník
>	vpravo orientovaný trojuholník
pentagram	päť cíp hviezda
hexagram	šesť cíp hviezda
none	žiadny (auto)

**Příklad 3: Vloženie inej funkcie do rovnakého grafu**

Základný graf si vytvoríme pomocou  $\sin(x)$ , ktorý sa nachádza v prílohe P II.

```
>>x = [0:0.05:1] * 2*pi;  
ysin = sin(x);  
plot(x,ysin'r');
```

```
>>hold on;                                     %Podržanie 1. grafu, aby sa mohol vložiť graf  
ycos = cos(x);                                   z druhej funkcie  
plot(x,ycos,'bx:');
```

**Příklad 4: Vyplnenie dvojrozmerného mnohoúhelníka**

Funkciou `fill (x,y,c)` je možné zobrazit farebný mnohoúhelník, kde  $x,y$  sú vektory súradníc bodov mnohoúhelníka a  $c$  je premenná, ktorá definuje farbu.

```
>>fill([0,1,0.5],[0,0,1],'g')
```

Graf sa nachádza v prílohe P II.

**Příklad 5: Nastavenie rozmerov osí x, y**

```
>>axis([0,100,0,100])
```

Graf sa nachádza v prílohe P II.

**Příklad 6: Nastavenie rozmerov osí x, y a z**

```
>>axis([0,10,0,10,1,5])
```

Graf sa nachádza v prílohe P III.

**Příklad 7: Rozmery osí sa nastavujú automaticky**

```
>>axis auto
```

**Příklad 8: Vypnutie zobrazenia osí**

```
>>plot([0:0.01:pi],sin([0:0.01:pi]))  
>>axis off
```

Graf sa nachádza v prílohe P II.

**Příklad 9: Zapnutie zobrazenia osí**

```
>>axis on
```

**Príklad 10: Rozdelenie grafického okna na subokná**

```
>>subplot(1,2,1)
plot(x,y,'r')
subplot(1,2,2)
plot(x,z,'b')
```

Graf sa nachádza v prílohe P II.

**Príklad 11: Popis grafu**

```
>>t = 0:0.01:2*pi;           % Definovanie premenných
y = sin(t);
plot(t,y);                   % vykreslenie
xlabel('t = 0 \rightarrow 2\pi'); % Popis osi x
ylabel('sin(t)');            % Popis osi y
title('Priebeh funkcie sínus') % Názov grafu
```

Graf sa nachádza v prílohe P II.

**Príklad 12: Legenda grafu**

```
y1 = sin(t); y2 = cos(t); y3 = y1+y2;
plot(t,y1,t,y2,t,y3);
legend('y_1=sin(t)', 'y_2=cos(t)', 'y_3=sin(t)+cos(t)')
```

Graf sa nachádza v prílohe P II.

**Príklad 13: Zobrazenie trojrozmerného obrazca**

```
>>[x,y] = meshgrid([-2:0.1:2]);
z = x.*exp(-x.^2-y.^2);
plot3(x,y,z);
```

Graf sa nachádza v prílohe P III.

**12.2 Najpoužívanéjšie grafy v oblasti automatizácie a teórie systému**

Medzi najpoužívanéjšie grafy v oblasti automatizácie a teórie systémov patria grafy pre impulzovú a prechodovú charakteristiku.

**12.2.1 Impulzová charakteristika**

$$Y(s) = G(s) \cdot U(s)$$

$$Y(s) = \frac{-3 \cdot s + 8}{s^2 + 4 \cdot s + 1} \cdot 1 = Y(s) \rightarrow y(t) = L^{-1}\{Y(s)\}$$

$$\begin{aligned}
y(t) &= \sum \operatorname{res} \left[ \frac{-3.s + 8}{(s + 2 - \sqrt{3}).(s + 2 + 3)} \cdot e^{st} \right]_{s=s_k} = \\
&= \operatorname{res} \left[ \frac{-3.s + 8}{(s + 2 - \sqrt{3}).(s + 2 + 3)} \cdot e^{st} \right]_{s=-2+\sqrt{3}} + \operatorname{res} \left[ \frac{-3.s + 8}{(s + 2 - \sqrt{3}).(s + 2 + 3)} \cdot e^{st} \right]_{s=-2-\sqrt{3}} = \\
&= \lim_{s \rightarrow -2+\sqrt{3}} \left[ \frac{4.s + 2}{(s + 2 + \sqrt{3})} \cdot e^{st} \right] + \lim_{s \rightarrow -2-\sqrt{3}} \left[ \frac{4.s + 2}{(s + 2 - \sqrt{3})} \cdot e^{st} \right] = \\
&= \frac{-3\sqrt{3} + 14}{2\sqrt{3}} \cdot e^{-2+\sqrt{3}t} + \frac{14 + 3\sqrt{3}}{-2\sqrt{3}} \cdot e^{-2-\sqrt{3}t} = \\
&= 2,542 \cdot e^{-2+\sqrt{3}t} - 5,542 \cdot e^{-2-\sqrt{3}t} = y(t)
\end{aligned}$$

```

>>x=0:0.01:100;
>>y=(2.542*2.71828.^(-0.268*x))-5.542*2.71828.^(-3.732*x));
impulse([-3 8],[1 4 1]);
hold on;
plot(x,y,'g--');

```

Graf sa nachádza v prílohe IV.

### 12.2.2 Prechodová charakteristika

$$Y(s) = G(s) \cdot U(s)$$

$$Y(s) = \frac{-3.s + 8}{s^2 + 4.s + 1} * \frac{1}{s} = Y(s) \rightarrow y(t) = L^{-1}\{Y(s)\}$$

$$\begin{aligned}
y(t) &= \sum \operatorname{res} \left[ \frac{-3.s + 8}{s.(s + 2 - \sqrt{3}).(s + 2 + \sqrt{3})} \cdot e^{st} \right]_{s=s_k} = \\
&= \operatorname{res} \left[ \frac{-3.s + 8}{s.(s + 2 - \sqrt{3}).(s + 2 + \sqrt{3})} \cdot e^{st} \right]_{s=0} + \operatorname{res} \left[ \frac{-3.s + 8}{s.(s + 2 - \sqrt{3}).(s + 2 + \sqrt{3})} \cdot e^{st} \right]_{s=-2-\sqrt{3}} + \\
&+ \operatorname{res} \left[ \frac{-3.s + 8}{s.(s + 2 - \sqrt{3}).(s + 2 + \sqrt{3})} \cdot e^{st} \right]_{s=-2+\sqrt{3}} = \lim_{s \rightarrow 0} \left[ \frac{-3.s + 8}{(s + 2 - \sqrt{3}).(s + 2 + \sqrt{3})} \cdot e^{st} \right] + \\
&= \lim_{s \rightarrow -2+\sqrt{3}} \left[ \frac{-3.s + 8}{(s + 2 + \sqrt{3}).s} \cdot e^{st} \right] + \lim_{s \rightarrow -2-\sqrt{3}} \left[ \frac{-3.s + 8}{(s + 2 - \sqrt{3}).s} \cdot e^{st} \right] = \\
&= \frac{8}{(+2 - \sqrt{3}).(+2 + \sqrt{3})} \cdot e^{0.t} + \frac{14 - 3\sqrt{3}}{2\sqrt{3} * (-2 + \sqrt{3})} \cdot e^{(-2+\sqrt{3})t} \\
&+ \frac{14 + 3\sqrt{3}}{-2\sqrt{3} * (-2 - \sqrt{3})} \cdot e^{(-2-\sqrt{3})t}
\end{aligned}$$

$$= 8 - 9,485 \cdot e^{(-2+\sqrt{3})t} + 1,485 \cdot e^{(-2-\sqrt{3})t} = y(t)$$

```
>>x=0:0.01:100;
y=8-(9.485).*2.71828.^(-0.268*x)-(1.485).*2.71828.^(-3.732*x);
step([-3 8],[1 4 1]);
hold on;
plot(x,y,'g--')
```

Graf sa nachádza v prílohe IV.

### 12.2.3 Nyquistova krivka

Dosadením reálnych hodnôt do zložkového tvaru získame Nyquistovu charakteristiku.

$$G(j \cdot \omega) = \frac{-20 \cdot \omega^2 + 8}{(-\omega^2 + 1)^2 + 16 \cdot \omega^2} + j \cdot \frac{3 \cdot \omega^3 - 35 \cdot \omega}{(-\omega^2 + 1)^2 + 16 \cdot \omega^2}$$

```
>>x=0:0.01:100;
re=(-20*x.^2+8)./(x.^4+14*x.^2+1);
im=(3*x.^3-35*x)./(x.^4+14*x.^2+1);
plot(re,im,'g--','lineWidth',3);
hold on;
nyquist([-3 8],[1 4 1])
```

Graf sa nachádza v prílohe IV.

### 12.2.4 Bodeho krivka

Exponenciálny tvar

$$G(j \cdot \omega) = \frac{\sqrt{9 \cdot \omega^6 + 190 \cdot \omega^4 + 905 \cdot \omega^2 + 64}}{(-\omega^2 + 1)^2 + 16 \cdot \omega^2} \cdot e^{j \cdot \arctan\left(\frac{3 \cdot \omega^3 - 35 \cdot \omega}{-20 \cdot \omega^2 + 8}\right)}$$

```
>>x=0:0.02:100;
y=20*log10(((9.*x.^6+190.*x.^4+905.*x.^2+64).^(1/2))./((-1.*x.^2+1).^2+16.*x.^2));
plot(x,y,'g--','lineWidth',3);
hold on;
bode([-3 8],[1 4 1]);
x1=0.67:0.02:100;
y1=atand((3.*x1.^3-35.*x1)./(-20.*x1.^2+8))+180;
plot(x1,y1,'g--','lineWidth',3);
x2=0:0.02:0.59;
y2=atand((3.*x2.^3-35.*x2)./(-20.*x2.^2+8))+360;
plot(x2,y2,'g--','lineWidth',3)
```

Graf sa nachádza v prílohe IV.

### 12.3 Neriešené príklady

1. Vytvorte graf funkcie  $\cos(3*t)$ , kde  $t = 0:0.1:\pi$
2. Vytvorenému grafu zmeňte farbu čiary na zelenú
3. Takto upravenému grafu zmeňte typ čiary na „---“ a vyznačte pomocou znaku X dôležité body
4. Vytvorte graf funkcie  $\sin(3*t)$ , ktorý bude červenej farby, a spojte ho s grafom funkcie  $\cos(3*t)$
5. K vytvorenému grafu funkcie  $\cos(3*t)$  doplňte názov grafu a popis jednotlivých osí

## 13 FUNKCIE

Ak sa nejaká postupnosť príkazov (algoritmus) opakuje vo viacerých situáciách (napr. pre rôzne hodnoty premenných), je lepšie využívať funkcie ako skripty, pretože u skriptov sa musia vždy upraviť hodnoty premenných, uložiť príslušný m-súbor a skript spustiť. Riešenie ponúkajú funkcie.

Funkcie sú m-súbory, ktoré majú presne definovanú štruktúru (viď. Vytvorenie funkcie). Funkcie akceptujú vstupné parametre, ktoré môžu mať pri každom spustení inú hodnotu.

Každá funkcia má svoje vlastné pracovné prostredie, ktoré je oddelené od pracovného prostredia Command Window. Všetky premenné vo funkcii sú lokálne (existujú iba vnútri funkcie). To znamená, že:

- sa nemôžu použiť žiadne iné premenné než tie, ktoré sú funkcie predávané (pomocou vstupných parametrov) alebo tie, ktoré si funkcia sama vytvorí,
- všetky premenné (aj tie, ktoré obsahujú vypočítané výsledky) po skončení funkcie zaniknú.

Našťastie existuje spôsob, ktorým funkcia môže svoje výsledky predať "von": výstupné premenné. Počet vstupných aj výstupných parametrov funkcie sa určuje pri jej vytváraní. Pokiaľ funkcia nemá žiadne vstupné parametre, môžu sa jej príkazy napísať tiež ako skript. (UIAM FCHPT STU, 2015)

### Príklad 1: Priemerná hodnota vektora

```
function y = priemer(x)
% PRIEMER= stredna hodnota vektora.           % Nápoveda pre funkciu
% PRIEMER(X), kde X je vektor vrati strednu hodnotu elementov vektora
% Ak vstup nie je vektor, vrati chybu.
x=1:10;                                         % Vygenerovanie vektoru
[m,n] = size(x);                               % Veľkosť vektoru
if ~(m == 1 | n == 1) | (m == 1 & n == 1)      % Podmienka, ktorá zisťuje či sa
jedná
                                                o vektor
error('Vstup musi byt vektor')                % Chybové hlásenie pokiaľ sa nejedná
                                                o vektor
end                                              % Koniec cyklu
y = sum(x)/length(x); % Výpočet               % Výpočet priemernej hodnoty vektora
```



**Príklad 2: Sčítanie dvoch čísiel**

```
function [s]=sucet(a,b)
% SUCET - sucet dvoch cisel
% s=sucet(a,b)
% a,b ... scitance
% s ... vysledok (sucet)
% priklad volania: s=sucet(10,-2.5)
if nargin~=2                                     % nespravny pocet vstupov?
    error('CHYBA: k vypoctu potrebujem DVA vstupy!') %Vypis chyby a konec funkcie
end % koniec if
s = a+b; % vypocet vysledku
```

**Príklad 3: Kvadratická rovnica**

```
function [x]=Kvadraticka(a,b,c)
% Vsetky riesenia kvadratickej rovnice  $a*x^2+b*x+c=0$ 
% [x]=Kvadraticka(a,b,c);
% x ... vektor vsetkych reseni
% a ... kvadraticky clen
% b ... linearny clen
% c ... absolutny clen
if a==0      % ak je to len lineárna rovnica
    x=Linearny(b,c);
else
    x=KvadratRed(b/a,c/a);          % Vydelenim ziskame redukovany tvar
end
```

**Príklad 4: Funkcia na zistenie typu premennej**

```
function vektor(x)
r=size(x);
if (r(1,1)==1) & (r(1,2)==1)
    disp('Promenna je skalar')
elseif length(r) >=2
    if ((r(1,1)==1) & (r(1,2)>=2)) / ((r(1,2)==1) & (r(1,1)>=2))
        disp('Promenna je vektor')
    else
        disp('Promenna je matice')
    end %if ((r(1,1)==1) & (r(1,2)>=2)) / ((r(1,2)==1) & (r(1,1)>=2))
end %if length(r)==1
```

**Príklad 5: Výpočet objemu, obsahu a uhlopriečky kvádra**

Hlavný program, ktorý môže byť uložený v inom M-súbore, alebo je napísaný priamo v príkazovom riadku:

```
clear all
a = input('zadej a: ');
b = input('zadej b: ');          % Zadanie parametrov
```

```
c = input('zadej c: ');  
[V,S,u] = kvader(a,b,c);
```

*% Volanie funkcie pre výpočet V, S, u kvádra*

```
function [V,S,u]=kvader(a,b,c) % Funkcia pre výpočet V, S, u kvádra  
% Vypocet objemu, povrchu a telesovej uhlopriecky  
% a, b, c ..... zadane hrany kvadra  
% V, S, u ..... objem, povrch, teles. uhlopriecka  
V = a*b*c;  
S = 2*(a*b+a*c+b*c);  
u = sqrt(a^2+b^2+c^2);
```

Výsledkom je:

V =  
125

S =  
150

u =  
8.6603

## 14 DÁTOVÉ SÚBORY

V tejto kapitole bude vysvetlené programovanie dátových súborov.

### 14.1 Práca s adresármi a súbormi

V tejto podkapitole bude popísaná práca s adresármi a súbormi.

#### 14.1.1 Práca s adresármi

S adresárom, ale aj súbormi je možné manipulovať pomocou okna Matlabu – Current Directory. Niekedy však pri písaní skriptov sa potrebuje na manipuláciu s adresármi a súbormi funkcia. Výstupom z týchto funkcií sú textové reťazce prípadne štruktúry, ktoré obsahujú požadované informácie. Príkaz *pwd* a *cd* vracia reťazec s absolútnou cestou ku aktuálnemu pracovnému adresáru. Zmenu adresára vykonáme príkazom *cd*, kde sa zadá meno cieľového adresára

```
>>pwd
ans = /Users/user/Documents/MATLAB
>>cd software
>>cd
ans = /Users/user/Documents/MATLAB/software
```

Obsah adresára sa zistí príkazmi *ls*, alebo *dir*. Príkaz *ls* vráti pod Unix-ovými systémami znakový vektor mien súborov oddelených tabulátormi a medzerami. Pod Windowsom príkaz *ls* vracia maticu znakov. Každý riadok je meno súboru doplnené medzerami tak, aby sa zhodovalo s najdlhším menom (znakové matice musia mať rovnaký počet znakov v stĺpci). Príkaz *dir* pracuje podobne ako príkaz *ls* s rozdielom, že výsledok je vrátený ako štruktúra s nasledovnými údajmi:

- *name* – meno súboru (pole znakov),
- *date* – dátum zmeny (pole znakov),
- *bytes* – počet bytov vyhradených pre súbor (double),
- *isdir* – 1 ak je položka name súbor, 0 ak adresár (logical),
- *datenum* – dátum zmeny ako MATLAB dátumové číslo (double).

```
>>adr = dir
>>adr(3)
ans =
name: 'data.dat'
date: '25-Apr-2015 18:41:39'
bytes: 19
```

*isdir: 0*

*datenum: 7.3419e+05*

Ďalšími príkazmi môžeme pracovať s konkrétnym adresárom. Príkaz *exist* kontroluje či existuje daný adresár. Ak áno, vráti ako návratovú hodnotu číslo 7, ak nie vráti 0. Zvyšné čísla sú vyhradené pre premenné a typy súborov. Zistíte ich pomocou príkazu *help exist*, alebo sa dajú nájsť v príručke. Príkazom *mkdir* sa vytvorí nový adresár. Ak použijeme jeden atribút vytvoríme adresár vzhľadom na aktuálny adresár. Adresár sa môže vytvoriť aj v inom adresári. V tom prípade bude prvý argument cesta ku adresáru, kde je potreba vytvoriť adresár s menom v druhom argumente funkcie. Adresár sa maže príkazom *rmdir*. Tu už neplatia pravidlá ako pri vytvorení adresára. Ak je potreba zmazať adresár aj s jeho obsahom je nutné pridať ako ďalší argument funkcie písmeno *s* ako reťazec.

```
>>exist('test')
```

```
ans = 0
```

```
>>mkdir('test')
```

```
% do adresara som mimo MATLAB vložil subor test.m
```

```
>>dir
```

```
. .. test.m
```

```
>>rmdir('test','s')
```

Príkaz	Popis
cd	zmena aktuálneho adresára
dir	výpis obsahu adresára
exist	zistuje existenciu adresára alebo súboru
fileattrib	nastavenie a získanie atribútov
filebrowser	otvorenie Current Directory
ls	výpis obsahu adresára
mkdir	vytvorenie adresára
pwd	cesta k aktuálnemu adresáru
rmdir	zmazanie adresára
what	zoznam súborov adresára podľa prípony

Obr. 15 Príkazy pre prevod pri práci s adresármi

(Blaho, 2009)

### 14.1.2 Práca so súbormi

Príkaz *edit* otvorí existujúci M-súbor na editáciu prípadne otvorí prázdny editor na tvorbu nového súboru. Súbor sa dá skopírovať zo zdrojovej adresy na cieľovú pomocou príkazu *copyfile*. Aj keď názov naznačuje, že ide o kopírovanie súborov, možno pomocou tohto príkazu kopírovať aj adresáre. Presunúť súbor (ale aj adresár) možno pomocou príkazu *movefile*. Funkcia s jedným argumentom prekopíruje zdroj do aktuálneho adresára, pridaním ďalšieho argumentu špecifikujeme cieľový adresár.

```
>>mkdir('test')
>>copyfile('test1.m','./ test ')
>>movefile('test2.m','./ test ')
>>dir('./ test ')
.  ..  test1.m  test2.m
>>movefile('./ test /test2.m','.')
>>dir('./ test ')
.  ..  test1.m
```

Na odstránenie súboru slúži príkaz *delete*. Cesta k súboru je absolútna alebo relatívna. Môže obsahovať náhradné znaky (wildcards ako je napr. \*). Príkaz *recycle* zistí, či sa odstránené súbory zmažú úplne alebo skončia v „odpadovom koši“. Vráti off ak ich Matlab zmaže úplne a on ak ich ukladá do systémového koša. Požadovanú hodnotu mu nastavíte aj sami ako textový reťazec (on/off).

```
>>recycle
ans = off
>>recycle on
>>[pathstr,name,ext,versn]=fileparts('./test/test1.m')
pathstr = ./test
name = test1
ext = .m
versn = ''
>>delete('./test/test*.*')
%kos je zapnuty mal by obsahovat zmazane subory
```

## 14.2 Import a Export do MAT-súborov

MAT-súbory sú binárne súbory s dvojitou presnosťou. Predstavujú štandardný formát súborov na ukladanie dát v Matlabe. Dáta uložené v MAT-súbore je možné prenášať a čítať na ďalšom počítači s rovnakou (často aj s rôznou) verziou MATLABu.

### 14.2.1 Export do MAT- súborov

Na exportovanie premenných z Workspace do binárneho súboru sa využíva funkcia *save*. Za menom funkcie sa uvádza meno súboru. Ak sa meno nezadá Matlab uloží dáta do súboru *matlab.mat*. Za menom súboru vymenúvame premenné, ktoré chceme uložiť. Ak nešpecifikujeme premenné Matlab uloží do súboru všetky premenné Workspace.

```
>>save menosubora  
>>save menosubora prem1 prem2 ... premN
```

Ak je potreba uložiť premenné, ktoré sa začínajú rovnakými znakmi nemusíme ich jednotlivovo vymenovať. Stačí ak sa použije špeciálny znak (wildcard character) \*. Obsah súboru bez jeho importu je možné pozrieť príkazom *whos* s prepínačom *file*, za ktorým nasleduje meno súboru.

```
>>save menosubora prem*  
>>whos -file menosubora
```

Matlab umožňuje uložiť štruktúru ako jedinečnú premennú, alebo uložiť jednotlivé položky štruktúry ako samostatné premenné v súbore. Ak je potreba položky uložiť samostatne musíme použiť prepínač *struct*.

```
>>S.a=10.0;  
>>S.b='Posterus';  
>>save mojastruktura.mat S  
>>save mojedata.mat -struct S  
>>save mojedata.mat -struct S b
```

Na prvých dvoch riadkoch je vytvorená jednoduchá štruktúra. Na treťom riadku sa do dátového súboru uložila celá štruktúra. Na ďalšom sa uložila do súboru štruktúra, ale ako jednotlivé položky. Posledný príkaz Matlabu povedal, že má uložiť iba položku *b* zo štruktúry *S*. Do dátového súboru sa môžu pridávať ďalšie premenné. Je to možné pomocou prepínača *append*. Podľa toho či sa už daná premenná nachádza v súbore alebo nie MATLAB vykoná nasledovné operácie:

- premenná už v súbore je – Matlab prepíše uložené údaje aktuálnymi vo Workspace,
- premenná v súbore nie je – Matlab pridá premennú a uloží jej údaje z Workspace.

### 14.2.2 Import z MAT- súborov

Načítanie dát zo MAT-súborov do Workspace je pomocou príkazu *load*. Väčšina syntaxe je rovnaká ako pri exporte. Ak je potreba načítať celý súbor do Workspace uvedie sa len

jeho meno. Podobne ako pri exporte je možné importovať vybrané premenná, prípadne využiť špeciálny znak `*`.

```
>>load menosubora  
>>load menosubora prem1 prem2 ... premN  
>>load menosubora prem*
```

Pri využití príkazu *whos* s prepínačom *file* a menom súboru uvidíme informácie ako sú mená premenných, ich rozmer alebo triedu (typ) premennej. Špecifikovaním výstupnej premennej funkcie *load* sa určuje premenná do ktorej sa načítajú všetky dáta naraz. Táto premenná bude logicky typu *struct*.

```
>>whos -file menosubora.mat  
>>S = load('menosubora.mat')
```

Pri čítaní textových (ASCII) dát musia byť dáta uložené s rovnakým počtom dát v riadku (inak hlási Matlab chybu). Dáta by mali byť oddelené prázdny miestom, čiarkou alebo tabulátorom. Počet riadkov premennej sa bude rovnať počtu riadkov v súbore. Meno premennej sa bude štandardne rovnať menu súboru, z ktorého dáta čítame bez prípony. Ak by sa súbor začínal podčiarkovníkom alebo číslom Matlab doplní pred premennú písmeno *X*.

### 14.3 Import textových súborov

Na import textových dát z príkazového riadka alebo M-súboru sa musí zavolať niektorá z MALABových funkcií pre import. Výber funkcie zväčša závisí od toho, ako sú dáta v súbore naformátované. Textové dáta musia byť naformátované do rovnakého počtu riadkov a stĺpcov. Ako oddelovač dát slúži oddelovací znak. Oddelovací znak môže byť medzera, čiarka, tabulátor alebo iný znak. Dáta môžu byť v súbore číselné prípadne znakové. Niektoré súbory môžu obsahovať tzv. hlavičky ku dátam na ich označenie.

Ak sú dáta uložené v súbore do obdĺžnika (rovnaký počet stĺpcov v riadku) najjednoduchším spôsobom ako ich načítať je s využitím funkcie *load*. Jednotlivé prvky by mali byť oddelené medzerou. Dáta sú importované do Workspace s rovnakým menom ako je meno súboru bez koncovky.

Pre určenie mena súboru, pod ktorým sa dáta uložia je potreba použiť nasledujúci príkaz.

```
>>load data.txt; alebo >>A = load('data.txt');
```

Pre iné oddelovače ako je medzera je možné využiť pokročilejšie príkazy ako je napríklad *dload*. Ako druhý parameter tejto funkcie sa uvádza oddelovač ako znak. Príkaz ignoru-

je medzery medzi dátami. Ako ďalšie parametre je možné uviesť posunutie dát, od ktorých sa má začať čítať (indexuje sa od nuly).

```
>>A = dlmread('data.txt', ';');  
>>A = dlmread('data.txt', ';', 2, 3);
```

Podobný príkaz ako *dlmread* je príkaz *csvread*. Tento príkaz sa využíva na čítanie dát z textových procesorov uložených v csv súboroch. V týchto súboroch sa dáta oddeľujú čiarkou preto tento znak nemusíme zadávať ako v príkaze *dlmread*. Zvyšné parametre zostávajú.

```
>>A=csvread('data.txt', 2, 3);
```

Niekedy textový súbor obsahuje hlavičky ku dátam, ktoré je potreba často vynechať. V takomto prípade sa využije funkcia *textscan*. V tejto funkcii sa môže zadať parameter *headerlines*, ktorý slúži na ignorovanie prvých N riadkov (kde sa nachádzajú hlavičky). Výstup príkazu *textscan* sa ukladá do premennej, v ktorej je pole typu *cell*.

```
Grade1 Grade2 Grade3  
78.8    55.9    45.9  
99.5    66.8    78.0  
89.5    77.0    56.7
```

Dáta zo súboru *data.txt* sa načítajú nasledovne:

```
>>fid = fopen('data.txt', 'r');  
>>grades = textscan(fid, '%f%f%f', 3, 'headerlines', 1);  
>>fclose(fid);
```

V tomto príklade sa najskôr otvorí súbor na čítanie, premenná *fid* obsahuje identifikátor súboru. Príkazom *textscan* sa načítali 3 riadky, pričom dáta boli načítané ako čísla s dvojitou presnosťou a bol vynechaný prvý riadok, kde sa nachádzajú hlavičky. Znak *%f* je typ dát a sú totožné so znakmi konverzie. Súbor musí byť uzavretý príkazom *fclose*.

Prepínače pre určenie typu dát zohrávajú dôležitú úlohu pri čítaní textových súborov, ktoré majú v riadku rôzny typ dát. Tieto dáta sa čítajú pomocou funkcií *textscan* a *textread*. Príkaz *textread* dokáže uložiť dáta do užívateľov špecifikovaných premenných. Funkcia *textscan* je rýchlejšia pri väčších súboroch. Použitie bude ukázané na nasledujúcich dátach, ktoré je aj v užívateľskej príručke.

```
Sally Type1 12.34 45 Yes  
Larry Type2 34.56 54 Yes  
Tommy Type1 67.89 23 No
```



Takéto dáta sa načítajú pomocou príkazu *textread* nasledovne:

```
>>[names, types, x, y, answer] = ...  
textread('data.txt', '%s %s %f %d %s', 3)
```

Príkaz rozdelí dáta v súbore do jednotlivých premenných pričom ich typy určujú prepínače a počet riadkov na spravovanie sa rovná poslednému parametru.

## 14.4 Export do textových súborov

Pre export dát do textových súborov má MATLAB vstavaných niekoľko funkcií. Výber funkcie závisí od množstva dát a formátu výstupného súboru. V minulej časti bola popísaná najjednoduchšia funkcia *save*. Funkcia slúži na ukladanie do binárnych MAT-súborov, ale aj na ukladanie do ASCII súborov a ako oddeľovač používa medzeru.

```
>>A = [1 2 3 4 ; 5 6 7 8];  
>>save data.txt A -ASCII
```

Pre pokročilejšie možnosti exportu je možné využiť príkaz *dlmwrite*. Príkaz umožňuje špecifikovať vlastný oddeľovač dát, výber len určitej časti dát z matice alebo vektora prípadne znak ukončenia riadku, ktoré sa líšia pre Windowsové a Unixové systémy. Príkaz nepridáva oddeľovač na konci riadku. Oddeľovač sa pridáva do príkazu ako znak (teda v úvodzovkách) pričom sa štandardne využíva čiarka. Začiatok dát v matici sa zadáva číslom riadku a stĺpca pričom index prvého prvku je 0,0.

```
>>A = [1 2 3 4 ; 5 6 7 8];  
>>csvwrite('csvdata.txt',A, 1, 2)
```

Ďalším príkazom na ukladanie dát do textového súboru je príkaz *diary*. Tento príkaz však ukladá výstup z príkazového riadku Matlabu.

```
>>diary data.txt  
>>A = [1 2 3 4 ; 5 6 7 8];  
>>A  
>>diary off  
>>type data.txt
```

Týmto príkazom vypíšeme obsah súboru data.txt do príkazového riadku.

## 14.5 Import a export XLS

MATLAB podporuje prácu so súbormi niektorých tabuľkových procesorov. Funkcie, ktoré sú spomenuté v kapitole, pracujú so súbormi tabuľkového procesora Microsoft Excel.

MALAB obsahuje aj funkcie, ktoré pracujú so súbormi tabuľkového procesora Lotus 123. Tie užívateľ nájde v príručke a sú veľmi podobné ako pre tabuľkový procesor Microsoft Excel.

#### 14.5.1 Informácie o súbore

Matlab obsahuje príkaz *xlsinfo*, ktorý zistí či sa jedná o súbor procesora Microsoft Excel. Príkaz vráti reťazec Microsoft Excel Spreadsheet ak je súbor čitateľný. V opačnom prípade vráti príkaz prázdny reťazec. Príkaz dokáže zistiť mená hárkov, ktoré súbor obsahuje.

```
>>[typ, desc] = xlsinfo('Zosit1.xls')
typ =
Microsoft Excel Spreadsheet
desc =
'Hárok1' 'Hárok2' 'Hárok3'
Pre porovnanie bolo vyskúšané precitať súbor procesora OpenOffice.
>> [typ, desc] = xlsinfo('Tabulka.ods')
typ =
''
desc =
Unreadable XLS file.
```

#### 14.5.2 Export do XLS súboru

Pre export dát do Excelu je možné využiť funkciu *xlswrite*. Pre najjednoduchšie použitie funkcie je nutné zadať meno súboru ako znakový reťazec a maticu s dátami. Ak sa súbor nenachádza v pracovnom adresári MATLAB ho vytvorí. Matica by mala byť rozmeru MxN pričom platia isté obmedzenia. Hodnota M musí byť menšia ako 65536 a hodnota N menšia ako 256. Matica by mala byť číselného, alebo znakového dátového typu, prípadne typu cell (zmiešané dátové typy). Dáta budú zapísané do prvého hárku na pozíciu A1.

Meno hárku je možné určiť ďalším parametrom v podobe znakového reťazca. Ak hárok neexistuje pridá sa na koniec hárkov. MATLAB pritom upozorní textom Warning: Added specified worksheet. Dáta z matice je možné uložiť do rôznych buniek v hárku. Bunky zadáme ako reťazec napríklad 'C2:H5', kde prvá bunka je ľavý horný a druhá bunka pravý dolný roh dát. Ak je zadaná len jedna bunka, bude to bunka ľavého horného rohu. Ak je rozmer oblasti na zápis menší Matlab zapíše iba dáta, ktoré sa do danej oblasti zmestia. Ak je naopak oblasť väčšia Matlab do prázdnych buniek zapíše text #N/A. Ak sa zápis podarí, Matlab vráti logickú 1 a v opačnom prípade logickú 0 spolu s chybovou správou.

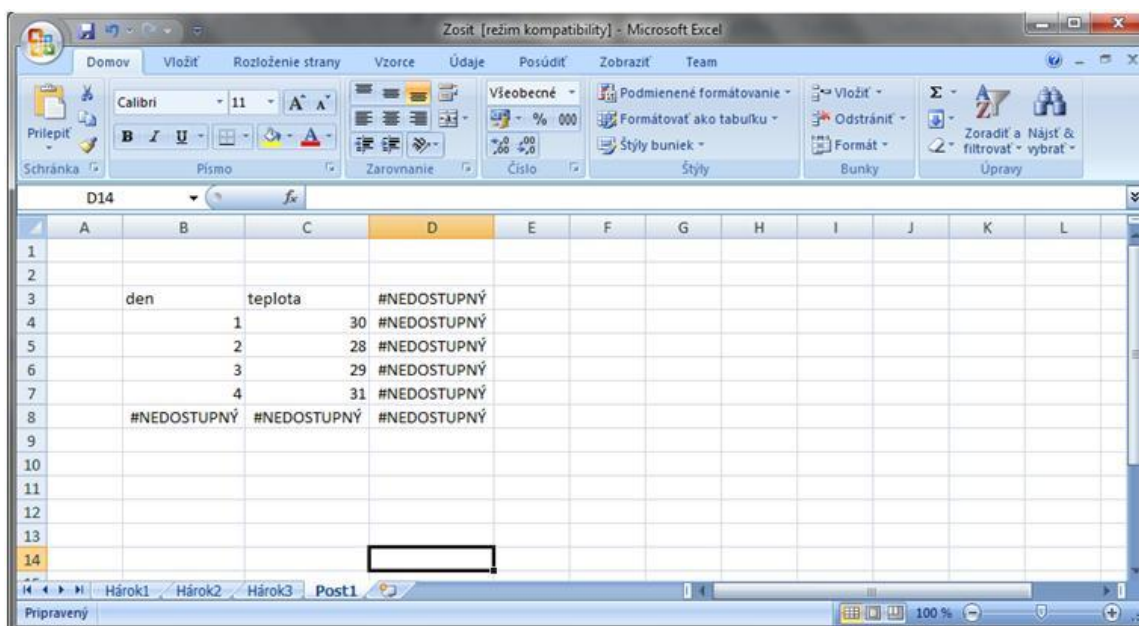
```
>>M={'den','teplota';1,30;2,28;3,29;4,31};
```

```
>>xlswrite('Zosit.xls',M);
>>xlswrite('Zosit.xls',M,'Post1')
Warning: Added specified worksheet.
```

Týmto spôsobom importujeme dáta od ľavého horného rohu. Ak chceme tabuľku posunúť na iné miesto, je potreba posledný príkaz zapísať takto:

```
>>xlswrite('Zosit.xls',M,'Post1','B3:D8')
```

Výstup príkazov môžete vidieť na Obr. 16. Znak #NaN je v slovenskej verzii označený ako #NEDOSTUPNÝ.



Obr. 16 Výstup príkazov pri exportu do XLS súboru  
(Vlastné spracovanie)

### 14.5.3 Import zo XLS súboru

Na čítanie dát z XLS súboru do Matlabu sa využíva príkaz *xlsread*. Funkcia je primárne určená na čítanie číselných dát. Preto po zadaní mena súboru vráti funkcia maticu čísel dátového typu double. Funkcia v základnom tvare ignoruje všetky nečíselné dáta mimo číselnej oblasti (napríklad popisy zľava alebo zhora). Ak sa v číselnej oblasti napriek tomu objavia nejaké nečíselné dáta sú nahradené číslom NaN.

Matlab umožňuje interaktívne vybrať dáta zo súboru. Stačí ako ďalší parameter zadať číselnú hodnotu -1. Matlab otvorí súbor, v ktorom sa označí dátová oblasť a voľba sa potvrdí stlačením tlačidla OK. Na nasledujúcom obrázku je vidno interaktívny výber dát. Dátum je v textovej podobe, a preto sa nebude nachádzať v numerickej časti výsledku príkazu.

```
>> num=xlsread('Databaza.xls',-1)
```

```
num =
```

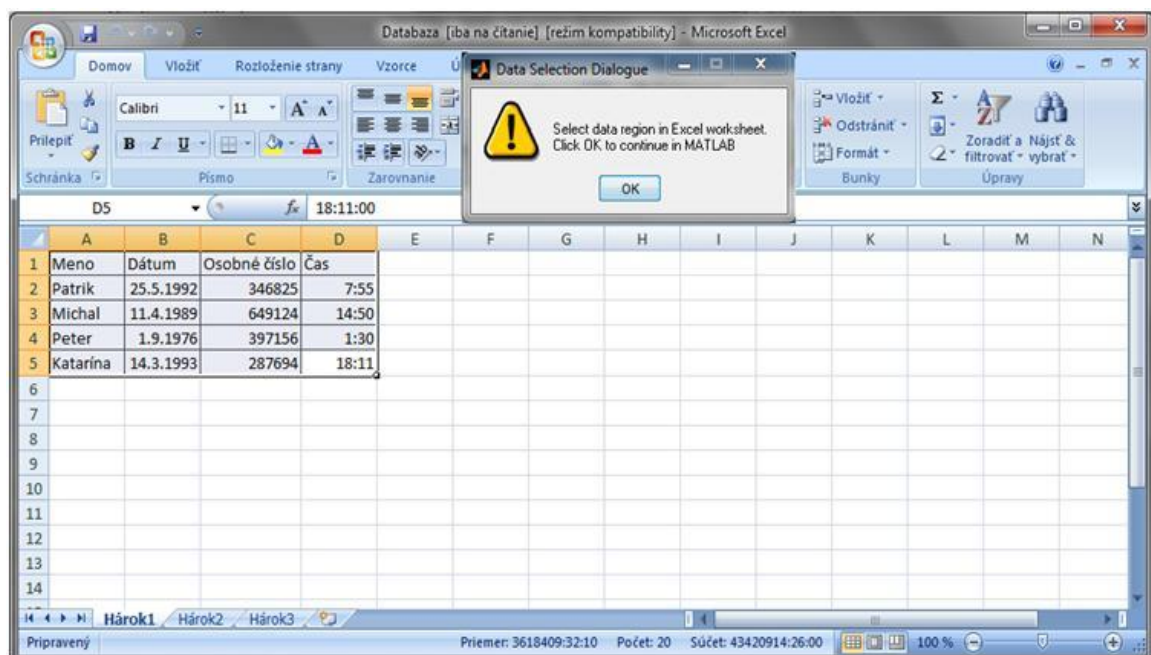
```
1.0e+05 *
```

```
3.4682 0.0000
```

```
6.4912 0.0000
```

```
3.9716 0.0000
```

```
2.8769 0.0000
```



Obr. 17 Výstup príkazov pri importe zo XLS súboru  
(Vlastné spracovanie)

Podobne ako pri príkaze na export je možné vybrať hárok a oblasť ako znakové reťazce. Mená hárkov si zistíte pomocou príkazu `xlsinfo`.

```
[num,txt]=xlsread('Databaza.xls')
```

```
num =
```

```
1.0e+05 *
```

```
3.4682 0.0000
```

```
6.4912 0.0000
```

```
3.9716 0.0000
```

```
2.8769 0.0000
```

```
txt =
```

```
'Meno'      'Dátum'      'Osobné číslo'  'Čas'
```

```
'Patrik '   '25. 5. 1992'  "                "
```

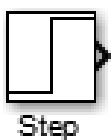
```
'Michal'    '11. 4. 1989'  "                "
```

```
'Peter'     '1. 9. 1976'   "                "
```

```
'Katarína'  '14. 3. 1993'  "                "
```

## 15 SIMULINK

Simulink je súčasť MATLABu, ktorá slúži ako nástroj pre modelovanie a analyzovanie dynamických systémov. Podporované sú lineárne i nelineárne systémy, spojité i diskrétné modely, prípadne ich kombinácie.



Blok *Scope* zobrazuje svoj vstup s ohľadom na simulačnú dobu.



Blok *Repeating Sequence* generuje na výstup periodický skalárny signál s definovaným priebehom.



Blok *Transfer Function* modeluje lineárne systémy pomocou Laplaceho transformácie.



Blok *Zero-Order Hold* vzorkuje a udržiava vstup pre určenú periódu vzorkovania.



Blok *Gain* násobí vstup o definovanú konštantnú hodnotu.



Blok *XY Graph* vykreslí zo vstupných hodnôt graf.



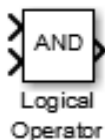
Blok *Sum* vykonáva sčítanie alebo odčítanie jeho vstupov.



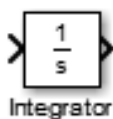
Blok *PID controller* je implementovaný PID regulátor, kde sa nastavujú hodnoty proporcionálnej, integračnej a derivačnej zložky.



Blok *Derivative* aproximuje derivát jeho vstupu výpočtem, kde  $du$  je zmena vstupných hodnôt a  $dt$  zmena v čase od predchádzajúceho kroku simulácie.



Blok *Logical Operator* je logický operátor, ktorý vykonáva zadanú logickú operáciu s jeho vstupnými impulzmi. Vstupná hodnota môže byť TRUE (log. 1) alebo FALSE (log. 0). V nastavení tohto bloku je možné vybrať inú logickú operáciu (OR, XOR, NOR, NAND, NXOR, NOT).



Blok *Integrator* integruje vstup v jeho súčasnom čase kroku.



Blok *Mux* kombinuje svoje vstupy do jedného vektora výstupu

## 15.1 Jednoduché príklady v Simulinku

V tejto podkapitole budú zobrazené ukážkové príklady práce v Simulinku.

**Príklad 1:** Namodelujte systém zobrazujúci rovnicu kružnice s polomerom  $r$ , daný rovnicami:

$$\begin{aligned}x &= r \sin(t) \\ y &= r \cos(t)\end{aligned}$$

Schéma sa nachádza v prílohe V.

**Príklad 2:** Namodelujte systém zobrazujúci logaritmickú špirálu:

$$\begin{aligned}x &= e^{-kt} \sin(t) \\ y &= e^{-kt} \cos(t)\end{aligned}$$

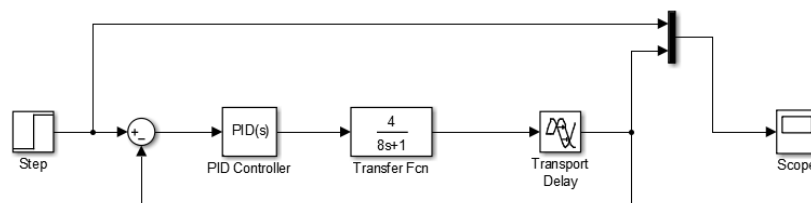
Schéma sa nachádza v prílohe V.

**Príklad 3:** Pre zadanú logickú funkciu  $y$  zostrojte schéma v Simulinku:

$$y = \bar{a}_1 * a_2 * a_1 * \bar{a}_2$$

Schéma sa nachádza v prílohe V.

**Příklad 4:** Zostrojte model PID regulátora:



Obr. 18 Model PID regulátora (Vlastné spracovanie)

Do *Transfer Fcn* zadávame  $G_-(s)$  a do *PID Controller* jednotlivé zložky vypočítaného PID regulátora. Výsledkom PID regulátora je graf, nachádzajúci sa v prílohe V.

## 16 GRAFICKÉ ROZHRAŇIE V MATLABE A SIMULINKU

Graphical user interface (GUI) je grafické rozhranie, ktoré obsahuje nástroje, alebo komponenty umožňujúce užívateľovi zdokonaľovať interaktívne úlohy. Zdokonalenie týchto úloh spočíva v tom, že užívateľ nemusí vytvárať program, ani písať jednotlivé príkazy do príkazového riadku. Často užívateľ ani nemusí vedieť detaily prepisu do programovacieho jazyka. Komponentmi GUI môže byť menu, nástrojový panel, príkazové tlačidlá, zaškrŕavacie políčka, zoznam a posuvník. V tomto prostredí je možné zobrazovať dáta aj v tabuľkovej forme. Vytvorenie nového grafického rozhrania umožňuje zadanie príkazu gui do príkazového riadku v základnom prostredí MATLAB.

Každý objekt v tomto prostredí je spojený s jednou, alebo viacerými procedúrami, nazývanými ako callback. Vykonanie každého takéhoto príkazu je spojené s čiastočnou užívateľskou aktivitou, ako napr. pri príkazovom tlačidle je to kliknutie myškou na dané tlačidlo. Táto časť programovania je často označovaná ako udalostné programovanie. Udalosťou je napr. kliknutie myškou na tlačidlo.

### 16.1 Aplikácia pre podporu výučby programu MATLABu

Aplikácia bola vytvorená pre podporu výučby v programe MATLAB. Aplikácia je zložená z viacerých kapitol. Každá kapitola obsahuje množstvo základných príkladov, ktoré študent využije pri výučbe. Grafické rozhranie aplikácie predstavuje akoby prezentáciu príkazov využitelných pri programovaní v MATLABe. Každá kapitola obsahuje príkazy a vysvetlenie jednotlivých príkazov, ktoré je rozdelené na viaceré slidy, aby aplikácia bola prehľadná. Každý slide obsahuje popmenu, kde si študent vyberie jednu z možností, o ktorú sa chce práve zaujímať. Po výbere jednej z možností sa na slide zobrazí ako vyzerá zápis príkazu do programu, a to buď do príkazového riadku, alebo do M-súboru. Ďalej sa tam nachádza výsledok zobrazeného príkazu. Poslednou položkou na slide je v ľavom dolnom rohu poznámka k zobrazenému príkazu, pretože v niektorých prípadoch nie je hneď jasné aké hodnoty, alebo premenné, sú v zobrazenom príkaze použité.



### 16.1.1 Popis grafického rozhrania aplikácie

Po spustení danej aplikácia sa zobrazí hlavné menu. Hlavné menu je zložené z ôsmich tlačidiel, ktoré predstavujú jednotlivé kapitoly. Tlačidlá a názvy jednotlivých kapitol sú zobrazené na nasledujúcom Obr. 19.



Obr. 19 Hlavné menu (Vlastné spracovanie)

## Kapitola MATICE

Táto podkapitola sa otvorí ak užívateľ aplikácie v hlavnom menu využije tlačidlo „MATICE“. Slidy pozostávajú z hlavného nadpisu, ktorý slúži ako základná informácia o tom, čo sa na danom slide nachádza. Pod nadpisom sa nachádza dôležitá časť slidu, v ktorej si užívateľ vyberá jednu z možností, ktorú chce zobraziť. Po výbere z jednej možnosti sa na slide v ľavej časti zobrazí príkaz použiteľný v MATLABe, na pravej strane výsledok príkazu a v dolnej ľavej časti je krátky komentár k danému príkazu.

Ve spodnej časti každého slide sú tri tlačidlá, ktoré je možné využiť pri potrebe sa vrátiť o slide naspäť, prejsť na hlavné menu alebo sa posunúť o slide dopredu. Ak je aktuálny slide prvý alebo posledný, užívateľ po kliknutí na jedno z krajných tlačidiel bude presmerovaný naspäť na hlavné menu.

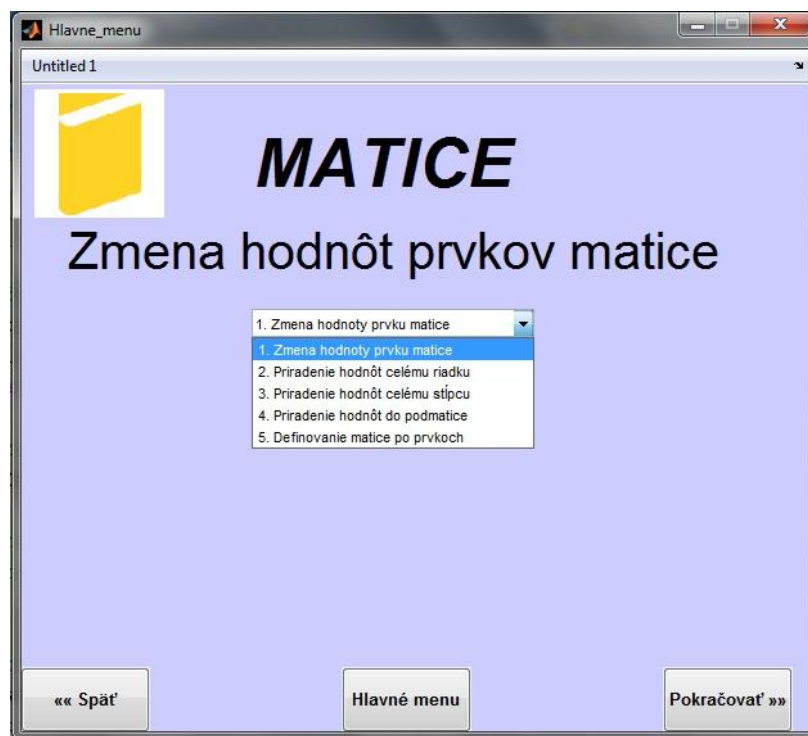
Na prvom slide sa nachádzajú popisy príkazov, ktoré slúžia na vytvorenie základných matic.



Obr. 20 Slide matic (Vlastné spracovanie)

Slide číslo 2 obsahuje základné operácie pre zmenu hodnoty prvku matice. Slide obsahuje príkazy, ktoré umožnia zmenu jednej hodnoty v matici, priradenie hodnôt celému riadku a stĺpcu, alebo určitej časti matice. Poslednou možnosťou je definovanie matice po jednotli-

vých prvkoch. Ako je vidieť z Obr. 21 grafické rozloženie tohto slide je rovnaké ako u predchádzajúceho slide.



Obr. 21 Slide zmeny hodnôt prvkov matice  
(Vlastné spracovanie)

Tretí slide obsahuje informácie o príkazoch, ktoré slúžia na vygenerovanie matice. Jedná sa o príkazy, ktoré slúžia na vygenerovanie základnej, štvorcovej, nulovej a matice zloženej z jednotiek. Grafické rozloženie tohto slide je opäť rovnaké ako je u predchádzajúcich slide, len s rozdielom iných príkazov.

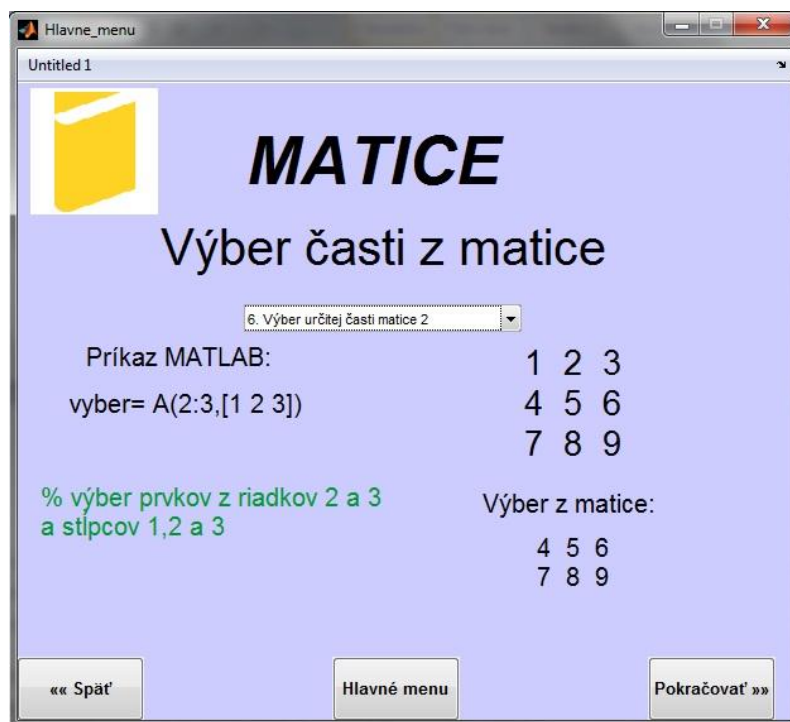
Na štvrtom a piatom slide sa nachádzajú príkazy, ktoré sa využívajú na spojenie matice a vektora, a to buď riadkového alebo stĺpcového. Grafické rozloženie tohto slide je opäť rovnaké ako je u predchádzajúcich slide, len s rozdielom iných príkazov.



Obr. 22 Slide spojenia matic a stĺpcového vektora  
(Vlastné spracovanie)

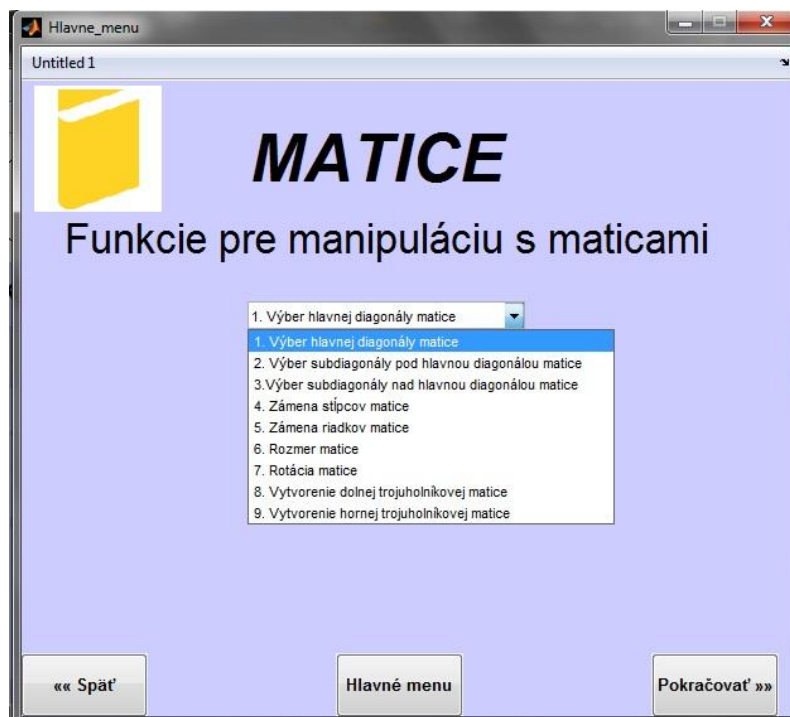
Slide číslo 6 obsahuje príkazy, ktoré sa využívajú pri spojení dvoch alebo viacerých matíc, či už vedľa seba alebo pod seba. Tento slide obsahuje taktiež ukážku príkazu, kde je možné spojiť viacerých matíc v kombinácii vedľa seba a pod seba.

Ďalší slide je zameraný na príkazy, ktoré sa využívajú na výber určitej časti z matice. Na tomto slide dochádza k malej zmene grafického rozloženia. V pravej hornej časti je zobrazená matica z ktorej vypisujeme určitú časť. Pod touto maticou sa nachádza výsledok, ktorý vznikne pri použití daného príkazu.

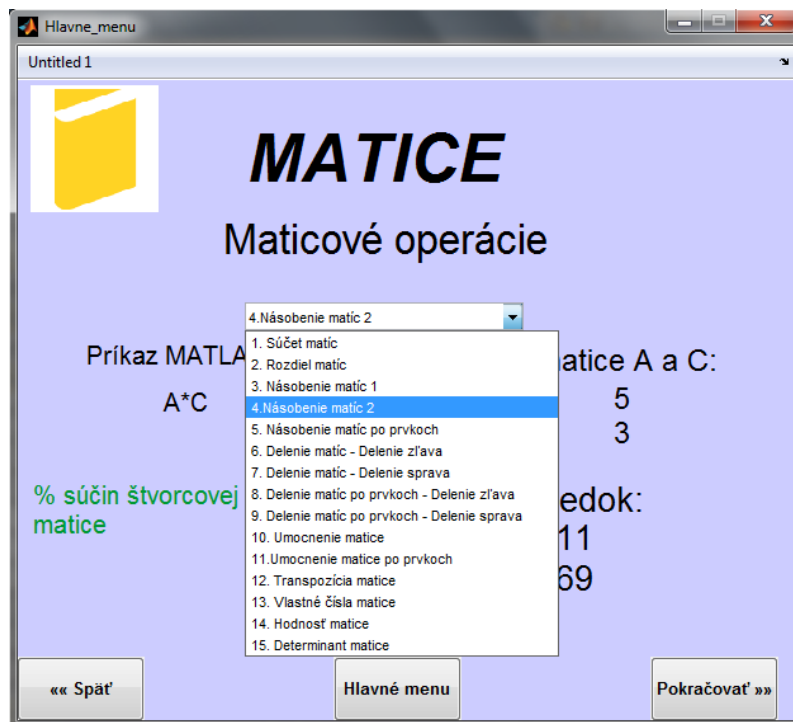


Obr. 23 Slide výberu časti z matice (Vlastné spracovanie)

Slide s poradovým číslom 8 obsahuje príkazy, ktoré slúžia pre manipuláciu s maticami.

Obr. 24 Slide funkcie pro manipuláciu s maticami  
(Vlastné spracovanie)

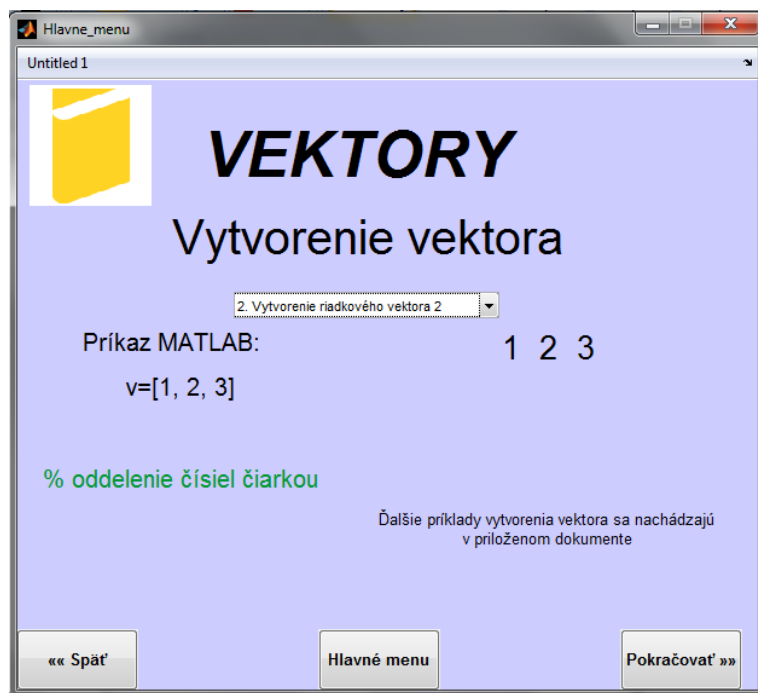
Posledný slide má poradové číslo 9. Na tomto slide sa nachádzajú základné maticové operácie ako súčet, rozdiel, súčin, delenie zľava, delenie sprava, umocnenie matice a mnoho ďalších. Celkový zoznam je možné vidieť na nasledujúcom obrázku. Grafické rozloženie tohto slide je rovnaké ako je u predchádzajúcich slide, teda obsahuje príkaz, výpis matic s ktorými príkaz pracuje, výsledok a stručný komentár k zadanému príkazu.



Obr. 25 Slide maticové operácie (Vlastné spracovanie)

## Kapitola VEKTORY

Táto kapitola obsahuje len jeden slide, v ktorom je ukážka príkazov ako je možné vektor vytvoriť. Túto kapitolu nebolo potreba ďalej rozširovať, pretože príkazy pre prácu s vektormi sú zväčša rovnaké ako pre matice.



Obr. 26 Slide vytvorenie vektora (Vlastné spracovanie)

Grafické rozloženie v kapitole vektory je rovnaká ako u matic. Obsahuje označenie kapitoly, hlavný nadpis, ktorý napovedá čo sa spustenom slide bude nachádzať za príkazy. ďalej slide obsahuje taktiež popmenu v ktorom si užívateľ vyberá jednu z možností, ktorú chce zobrazit'. Po výbere sa mu zobrazí príslušný príkaz, výsledok použitého príkazu a krátky komentár.

## Kapitola KOMPLEXNÉ ČÍSLA

Kapitola zameraná na komplexné čísla na prvom slide obsahuje príkazy, pomocou ktorých je možné komplexné číslo vytvoriť. Ako už bolo spomenuté v predchádzajúcich častiach popisu programu, obsahuje aj táto kapitola názov, hlavný nadpis, ktorý napovedá čo sa spustenom slide bude nachádzať za príkazy. Grafické rozloženie slide je taktiež rovnaké ako v predchádzajúcich kapitolách, teda obsahuje popmenu, kde užívateľ vyberá jednu z možností, ktorú chce zobraziť. Následne sa na slide zobrazí príkaz využiteľný v programe MATLAB, výsledok príkazu a krátky komentár.

Na druhom slide je sú príkazy, ktoré je možné využiť pri výpise samostatnej reálnej alebo imaginárnej časti komplexného čísla alebo absolútnej hodnoty.

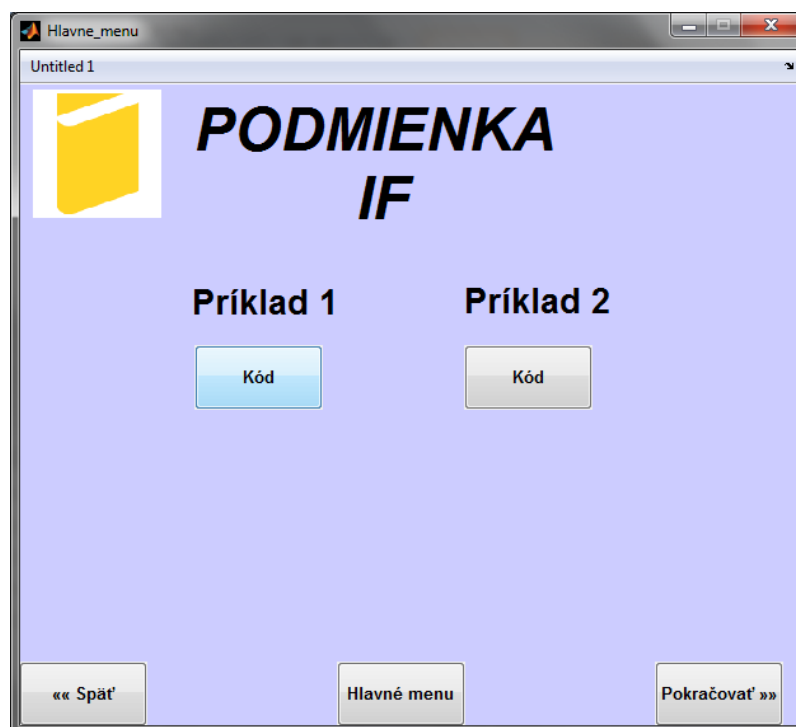


Obr. 27 Slide reálnej a imaginárnej časti  
(Vlastné spracovanie)



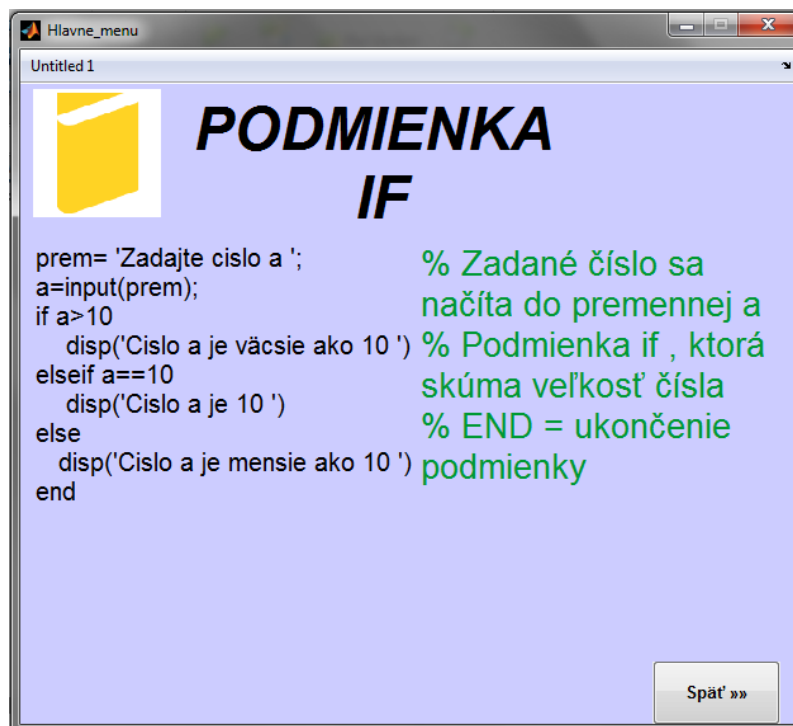
## Kapitola PODMIENKY A VETVENIE

Kapitola je zameraná na podmienku *if* a vetvenie pomocou *switch*. Prvý slide je zameraný na podmienku *if*. Grafické rozloženie tohto slide sa od doteraz popísaných slidu líši. Na základnom prvom slide je názov kapitoly, v tomto prípade je to podmienka *if*. Pod názvom tejto kapitoly sa nachádzajú dve tlačidlá, pomocou ktorých sa je možné presmerovať na ďalšie dva slide, kde sú kódy dvoch príkladov, v ktorých je podmienka *if* využitá.



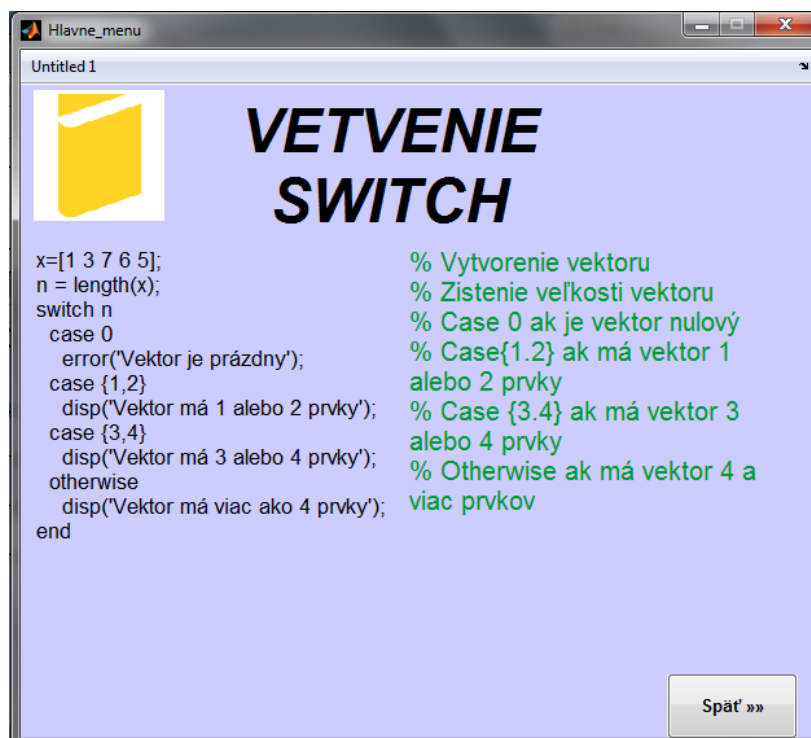
Obr. 28 Slide podmienky *If* (Vlastné spracovanie)

Na Obr. 29 sa nachádza ukážka príkladu číslo 2. Grafické rozloženie je v postate rovnaké, rozdiel je len v tlačidlách. Na tomto slide sa nachádza len jedno tlačidlo, ktoré užívateľa vráti späť na hlavný slide podmienky *if*. Každý príklad taktiež obsahuje komentár k príkladu.



Obr. 29 Slide podmienky If (Vlastné spracovanie)

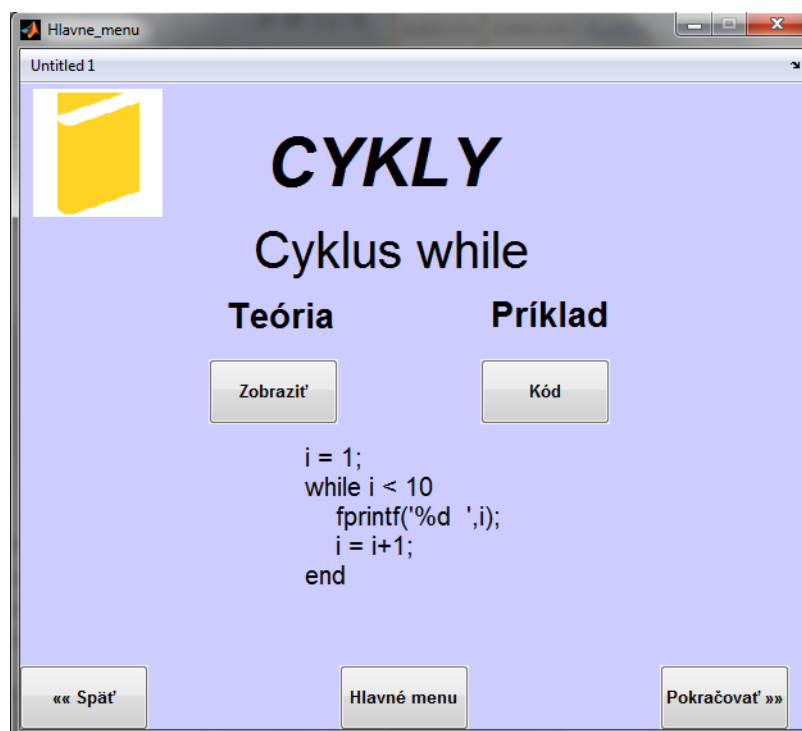
Druhý hlavný slide je zameraný na vetvenie pomocou *switch*. Grafické rozloženie je rovnaké ako u podmienky *if*. taktiež na hlavnom slide obsahuje dve tlačidlá, ktoré presmerujú užívateľa na jednotlivé príklady.



Obr. 30 Slide vetvenie switch (Vlastné spracovanie)

## Kapitola CYKLY

V tejto kapitole sú vysvetlené cykly *while* a *for*. Prvý slide je venovaný cyklu *while*. Štandardne obsahuje názov kapitoly, pod ktorým sa nachádzajú taktiež dve tlačidlá. Jedno je venované krátkej teórii a druhé je venované príkladu. Pod tlačidlami na znázornená syntaxia príslušného cyklu. Druhý slide je venovaný cyklu *for*, ktorého grafické rozloženie je rovnaké ako u cyklu *while*.



Obr. 31 Slide cyklov (Vlastné spracovanie)

Na nasledujúcom obrázku je zobrazená teória cyklu *while*, na ktorú je užívateľ presmerovaný po kliknutí na tlačidlo zobraziť na slide, ktorý je venovaný cyklu *while*.



Obr. 32 Slide cyklu while (Vlastné spracovanie)

Obr. 33 v tejto kapitole ukazuje príklad, v ktorom je využitý cyklus for. Graficky je na tom rovnako ako teória. Na ľavej strane sa nachádza kód s využitím cyklu for, a na pravej strane sú komentáry k tomuto kódu.



Obr. 33 Slide cyklu for (Vlastné spracovanie)

## Kapitola REŤAZCE

Táto kapitola sa skladá z jedného slide, ktorý je zameraný na základné operácie s reťazcami. Na tomto slide sa nachádza v popmenu 6 možností operácií s reťazcami ako je napríklad ich spojenie vedľa alebo pod seba. Na obrázku je vidieť že grafické rozloženie je rovnaké ako napríklad u kapitoly zameranej na matice alebo vektory.



Obr. 34 Slide reťazce (Vlastné spracovanie)

## Kapitola GRAFY

Táto kapitola je zameraná na vytvorenie základných grafov. Obsahuje ukážky ako základné grafy vytvoriť, zmeniť ich parametre ako je typ čiary, farba čiary a označenie bodov. Kapitola obsahuje aj ukážku ako viaceré grafy spojiť do jedného zobrazenia. Grafické rozloženie obsahuje názov kapitoly, hlavný nadpis, ktorý pomáha určiť čo sa na danom slide nachádza, následne kus kódu pomocou ktorého je následný graf možné zostrojiť, jednoduchý komentár a v dolnej časti tlačidlo „Zobraziť“, ktoré presmeruje užívateľa na slide, kde je popísaný graf vykreslený. Každý slide taktiež obsahuje trojicu tlačidiel, pomocou ktorých je možné sa vrátiť o jeden slide naspäť, posunúť o jeden slide dopredu, alebo sa vrátiť na hlavné menu.

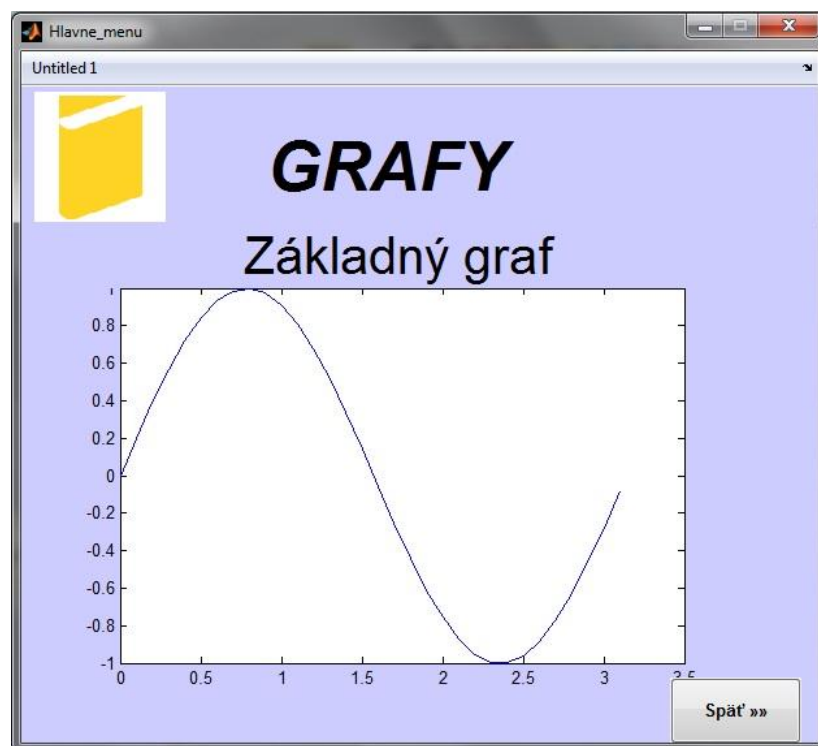


Obr. 35 Slide 2D grafy (Vlastné spracovanie)

Na Obr. 36 sa nachádza slide základného grafu



Obr. 36 Slide základný graf (Vlastné spracovanie)



Obr. 37 Slide zobrazenie grafu (Vlastné spracovanie)

## ZÁVER

MATLAB patrí k jednému z najpoužívanějších programovacích jazykov využívaných pre množstvo výpočtov. MATLAB je dnes využívaný predovšetkým pre svoju rozmanitosť, ktorú ponúka prostredníctvom širokej palety aplikácií, vďaka ktorým je schopný sa vyrovnáť veľmi podobným konkurenčným programom.

Hlavným cieľom práce bolo vytvoriť prehľad základných príkladov a navrhnuť výučbovú aplikáciu vhodnú pre študentov, ktorí začínajú s programom MATLAB a Simulink.

Práca oboznámi čitateľa s históriou daného programu, s podrobným popisom rozhrania programovacieho prostredia a jednotlivými prvkami, ktoré sa v programe MATLAB a Simulink využívajú napr. Práca s premennými, dátovými typmi, operátormi či podmienkami a cyklami.

Druhá časť práce je zameraná na praktickú stránku programu MATLAB, kde bol vytvorený prehľad príkladov a príkazov používaných v prostredí MATLAB. Boli vytvorené aj neriešené príklady, na ktorých si môže čitateľ overiť, či daný okruh dostatočne pochopil.

V závere práce bola vysvetlená tvorba interaktívnej aplikácie pre podporu výučby v programe MATLAB, ktorá ponúka výučbu v programe MATLAB prostredníctvom ukážky správneho zapísania príkazu do programovacej časti MATLABu.

Celkové spracovanie bolo orientované pre študentov, ktorý v tomto prostredí iba začínajú a nemajú s týmto programom skúsenosti, alebo prípadne minimálne.



**ZOZNAM POUŽITEJ LITERATURY**

- [1] BLAHO, Michal, 2007. *Matlab-komplexné čísla*. Posterus.sk: portál pre odborné publikovanie [online]. Roč. 7, č. 2 [cit. 2015-03-10]. DOI: ISSN 1338-0087. Dostupné z: <http://www.posterus.sk/?p=109>
- [2] BLAHO, Michal, 2009. *Matlab – znaky a textové reťazce*. Posterus.sk: portál pre odborné publikovanie [online]. Roč. 2, č. 6 [cit. 2015-05-04]. DOI: ISSN 1338-0087. Dostupné z: <http://www.posterus.sk/?p=672>
- [3] DUŠEK, František, 2000. *Matlab a Simulink: úvod do používání*. 1.vyd. Pardubice: Univerzita Pardubice, 146 s. ISBN 80-7194-273-1.
- [4] HUMUSOFT: Technické výpočty, řídicí technika, simulace ... *Humusoft.cz* [online]. ©1991-2015 [cit. 2015-02-13]. Dostupné z: <http://humusoft.cz/produkty/matlab/>
- [5] KARBAN, Pavel, 2006. *Výpočty a simulace v programech Matlab a Simulink*. Brno: Computer Press, a.s. ISBN 978-80-251-1448-3.
- [6] KOZÁK, Štefan a Slavomír KAJAN, 1999. *MATLAB-SIMULINK 1*. 1. vyd. Bratislava: Slovenská Technická Univerzita v Bratislavě, 125 s. ISBN 80-277-1213-2.
- [7] KUPKA, Libor, 2007. *MATLAB Simulink: úvod do používání*. Lanškroun. CZ.04.1.03/3.1.15.1/0005. Projekt. SOŠ a SOU Lanškroun.
- [8] MAJEROVÁ, Dana, 2013. UPŘT: Ústav počítačové a řídicí techniky. *Cykly* [online]. [cit. 2015-03-15]. Dostupné z: <http://uprt.vscht.cz/majerova/matlab/lekce6.html>
- [9] MathWorks: Company. *MathWorks.com* [online]. ©1994-2015 [cit. 2015-02-13]. Dostupné z: <http://www.mathworks.com/company/aboutus/>
- [10] PERŮTKA, Karel, 2005. *MATLAB: základy pro studenty automatizace a informačních technologií*. Vyd. 1. Zlín: Ústav řízení procesů, Institut řízení procesů a aplikované informatiky, Fakulta technologická, Univerzita Tomáše Bati ve Zlíně, 303 s. ISBN 80-7318-355-2.
- [11] UIAM FCHPT STU, 2015. Ústav informatizácie, automatizácie a matematiky: Fakulta technickej a potravinárskej technológie, Slovenská technická univerzita v Bratislavě [online]. Bratislava. [cit.2015-03-15]. Dostupné z: <https://www.kirp.chtf.stuba.sk/~cirka/vyuka/matlab/kap3.php#sec1>

- [12] WIRTH, Michael a Peter KOVESI, 2005. *MATLAB as an Introductory Programming Language*. Wiley Online Library [online]. Č. 1, s. 11 [cit. 2015-02-13]. Dostupné z: <http://onlinelibrary.wiley.com/doi/10.1002/cae.20064/pdf>
- [13] ZAPLATÍLEK, Karel a Bohuslav DOŇAR, 2004. *MATLAB - tvorba uživatelských aplikací*. Praha: BEN - technická literatura. ISBN 80-7300-133-0.

**ZOZNAM OBRÁZKOV**

Obr. 1 Dostupné rozšírenia Matlabu .....	14
Obr. 2 Odvetvia priemyslu s najčastejším využitím programu MATLAB a Simulink .....	15
Obr. 3 Pracovná plocha MATLAB .....	18
Obr. 4 Prostredie Workspace .....	19
Obr. 5 Prostredie Command History .....	20
Obr. 6 Nápoveda programu MATLAB .....	20
Obr. 7 Syntaxová nápoveda .....	21
Obr. 8 M-súbor v MATLABu .....	22
Obr. 9 Zobrazení Simulink Library Browser .....	25
Obr. 10 Grafické prostredie Simulink .....	26
Obr. 11 Subsystem .....	27
Obr. 12 Cyklus while .....	43
Obr. 13 Cyklus for .....	44
Obr. 14 Graf funkcie $\sin(2 \cdot t)$ .....	80
Obr. 15 Príkazy pre prevod pri práci s adresármi .....	91
Obr. 16 Výstup príkazov pri exportu do XLS súboru .....	98
Obr. 17 Výstup príkazov pri importe zo XLS súboru .....	99
Obr. 18 Model PID regulátora .....	102
Obr. 19 Hlavné menu .....	104
Obr. 20 Slide matic .....	105
Obr. 21 Slide zmeny hodnôt prvkov matice .....	106
Obr. 22 Slide spojenia matic a slúpcového vektora .....	107
Obr. 23 Slide výberu časti z matice .....	108
Obr. 24 Slide funkcie pro manipuláciu s maticami .....	108
Obr. 25 Slide maticové operácie .....	109
Obr. 26 Slide vytvorenie vektora .....	110
Obr. 27 Slide reálnej a imaginárnej časti .....	111
Obr. 28 Slide podmienky If .....	112
Obr. 29 Slide podmienky If .....	113
Obr. 30 Slide vetvenie switch .....	113
Obr. 31 Slide cyklov .....	114
Obr. 32 Slide cyklu while .....	115

---

Obr. 33 Slide cyklu for .....	115
Obr. 34 Slide reťazce .....	116
Obr. 35 Slide 2D grafy.....	117
Obr. 36 Slide základný graf .....	117
Obr. 37 Slide zobrazenie grafu .....	118

**ZOZNAM TABULIEK**

Tab. 1 Najpoužívanéjšie riadiace znaky .....	29
Tab. 2 Datové typy uint8,uint16, uint32, uint64.....	32
Tab. 3 Dátové typy int8, int16, int32, int64.....	33
Tab. 4 Unárne aritmetické operátory .....	38
Tab. 5 Binárne aritmetické operátor .....	38
Tab. 6 Špeciálne znaky .....	40
Tab. 7 Priorita operátorov .....	40
Tab. 8 Príkazy pre prevod medzi číselnými a znakovými typmi.....	78
Tab. 9 Grafické funkcie zobrazenia .....	81
Tab. 10 Grafické funkcie zobrazenia .....	81

**ZOZNAM PRÍLOH**

P I	Mnohonásobné vetvenie
P II	2D grafy
P III	3D grafy
P IV	Najpoužívanější grafy v oblasti automatizácie a teórie systému
P V	Schémy zo Simulinku
P VI	Aplikácia pre výučby v programe MATLAB na DVD
P VII	Príklady vytvorené v programe MATLAB na DVD

## PRÍLOHA P I: MNOHONÁSOBNÉ VETVENIE

### Príklad: Jednoduchá simulácia pošty: vstupného terminálu

```
a=0;b=0;c=0;d=0;e=0;f=0;g=0;r=0;t=0;z=0; % Deklarácia a vynulovanie premenných
                                         na poradové číslo

disp('-----');
disp('MENU');
disp('1. Poslanie listu');
disp('2. Platba');
disp('3. Vyzdvihnutie balíka, listu. ');
prem1= 'Vasa volba: ';
r=input(prem1); % Uloženie voľby z hlavného menu
                do premennej r

disp('-----');
switch r
case 1
    disp('1. Obyčajne ');
    disp('2. Doporučene'); % Menu pre voľbu 1
    prem2= 'Vasa volba: '; % Uloženie voľby z hlavného menu do
    t=input(prem2);        premennej t
    if t==1 % Podmienka pre zistenie výberu z
                podmenu
        a=a+1; % Iterácia pre premennú a
        if 1>a || a>9 % Kontrola premennej a, či sa
                        nachádza v správnom rozsahu
            a=1; % Ak nie, tak sa nastaví do premennej
                  a hodnota 1
        fprintf('Vase cislo je: %d \n',a); % Výpis hodnoty poradového čísla
    else
        fprintf('Vase cislo je: %d \n',a); % Výpis hodnoty poradového čísla ak
                                           nie je splnená podmienka if 1>a || a>9
    end
else % Ak neplatí t==1 vykoná sa else
    (t==2) % Rovnaký postup len z premennou b
    b=b+1;
    if 10>b || b>19
        b=10;
        fprintf('Vase cislo je: %d \n',b);
    else
        fprintf('Vase cislo je: %d \n',b);
    end
end
case 2
    disp('1. Platba do 5000');
    disp('2. Platba nad 5000');
    prem4= 'Vasa volba: ';
    u=input(prem4);
    if u==1
```

```

    e=e+1;
    if 29>e // e>39
        e=30;
        fprintf('Vase cislo je: %d \n',e);
    else
        fprintf('Vase cislo je: %d \n',e);
    end
else
    f=f+1;
    if 39>f // f>49
        f=40;
        fprintf('Vase cislo je: %d \n',f);
    else
        fprintf('Vase cislo je: %d \n',f);
    end
end
case 4
    g=g+1;
    if 49>g // g>59
        g=60;
        fprintf('Vase cislo je: %d \n',g);
    else
        fprintf('Vase cislo je: %d \n',g);
    end
end
end

```

V tomto príklade sa jedná o jednoduchú simuláciu terminálu na pošte, ktorý sa nachádza na vstupe, kde si zákazníci pošty vyberajú jednu z možností, ktoré potrebujú na pošte vybaviť. Každá z možností terminálu má iné hodnoty čísla, aby zamestnanci pošty boli pripravení na potreby zákazníka. Tento program obsahuje iné hodnoty čísel pre každý výber z hlavného menu a následne priradeného podmenu. Program obsahuje jednoduché ošetrenie, ktoré zabezpečuje aby nedošlo k prekrytiu medzi hodnotami čísel jednotlivých podmenu. Je to napríklad medzi obvyčajným poslaním listu, kde sú hodnoty priradených čísiel od 1 do 9, a poslaním doporučene, kde sú hodnoty od 10 do 19. Znamená to, že ak zákazník chce poslať list obvyčajným spôsobom, nemôže získať poradové číslo vyššie ako 9, čo by znamenalo, že terminál ho priradí k priehradke kde sa posielajú listy doporučene. Ak sa terminál pri pridelovaní čísiel dostane na najvyššiu povolenú hodnotu v jednotlivých častiach programu, nastaví hodnotu na najnižšiu a pokračuje od začiatku.

Terminály, ktoré sa nachádzajú na poštách, majú širšie číselné rady, čím sa zabráni tomu, že budú mať dvaja zákazníci rovnaké poradové číslo.

Spustenie M-súboru:



## *MENU*

*1. Poslanie listu*

*2. Platba*

*3. Vyzdvihnutie balíka, listu.*

*Vasa volba:*

*% Sem zadá užívateľ svoju voľbu z menu*

-----  
*1. Obvyčajne*

*% Podmenu pre voľbu 1z menu*

*2. Doporučene*

*Vasa volba:*

*% Tu zadá užívateľ svoju voľbu z podmenu*

*Vase číslo je:*

*% Výpis poradového čísla*

*1. Platba do 5000*

*2. Platba nad 5000*

*% Podmenu pre 3.. výber z menu*

*Vasa volba:*

*Vase číslo je:*

*% Výpis poradového čísla*

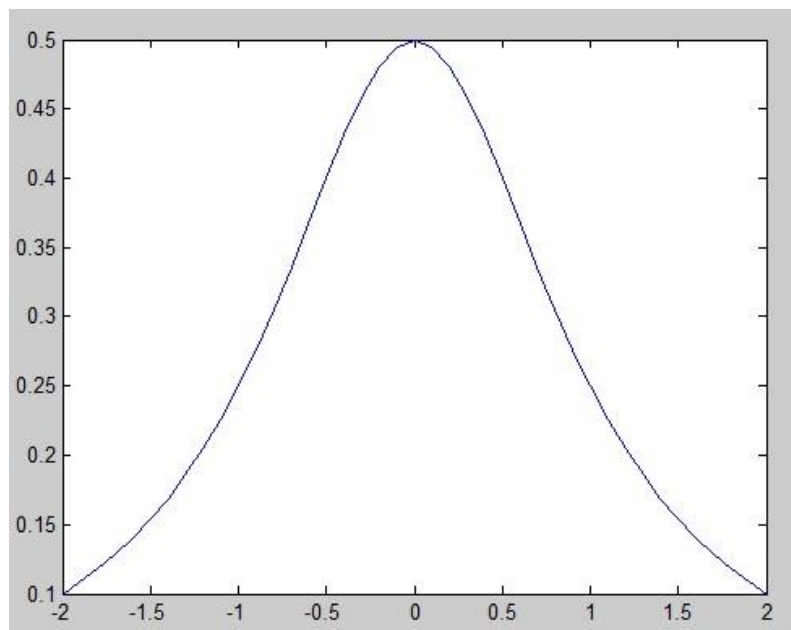
*Pri voľbe 4 z hlavného menu sa zobrazí len poradové číslo*

*Vase číslo je:*

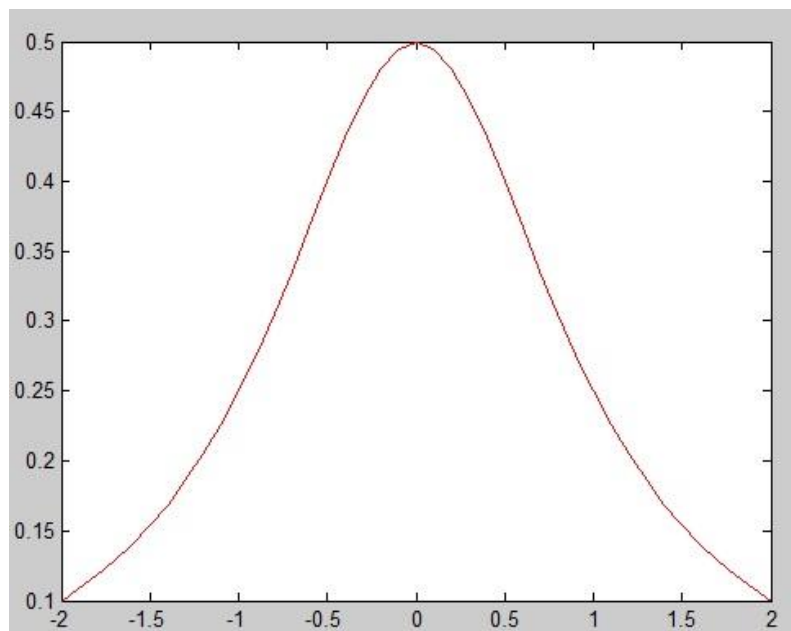
*% Výpis poradového čísla*

## PRÍLOHA P II: 2D GRAFY

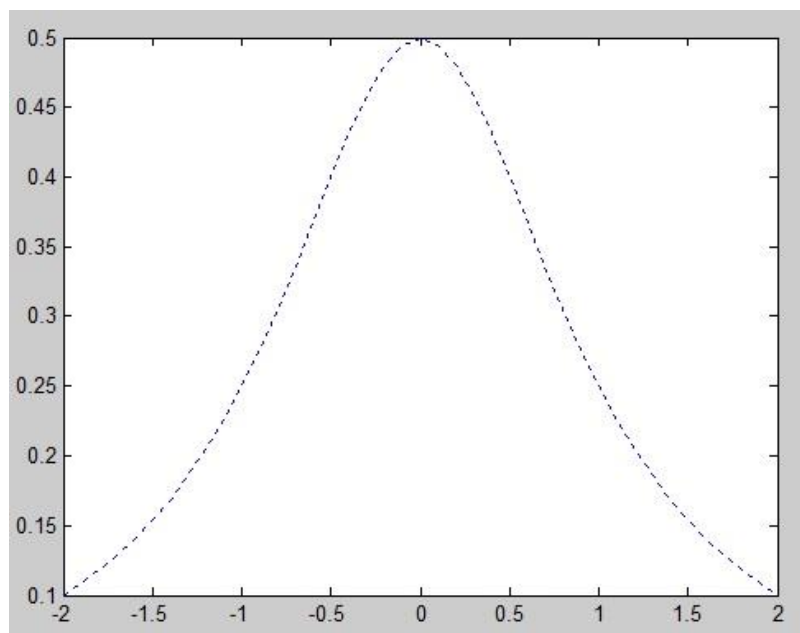
Graf funkcie  $1/(2+2x^2)$



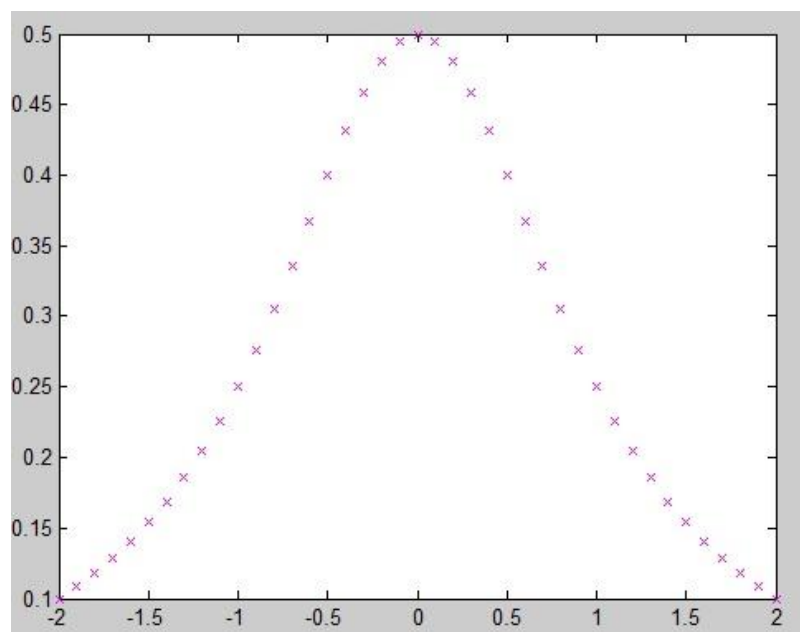
Graf funkcie  $1/(2+2x^2)$  pri zmene farby



**Graf funkcie  $1/(2+2x^2)$  pri zmene čiar**

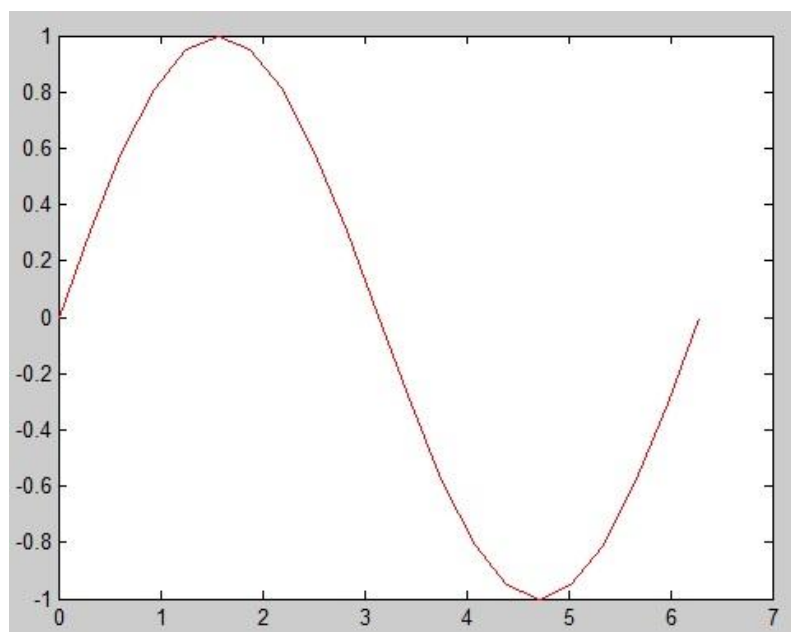


**Graf funkcie  $1/(2+2x^2)$  bez spojovania bodov**

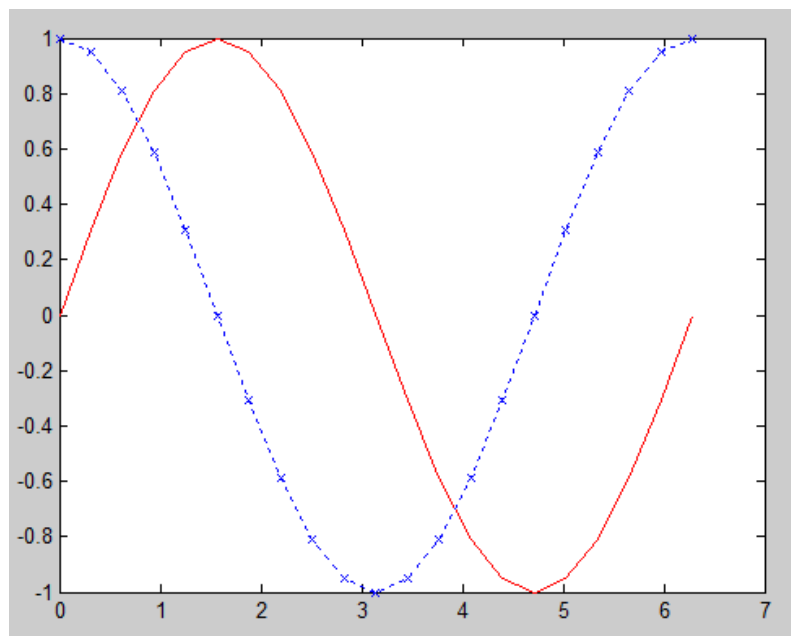


## Vloženie inej funkcie do rovnakého grafu

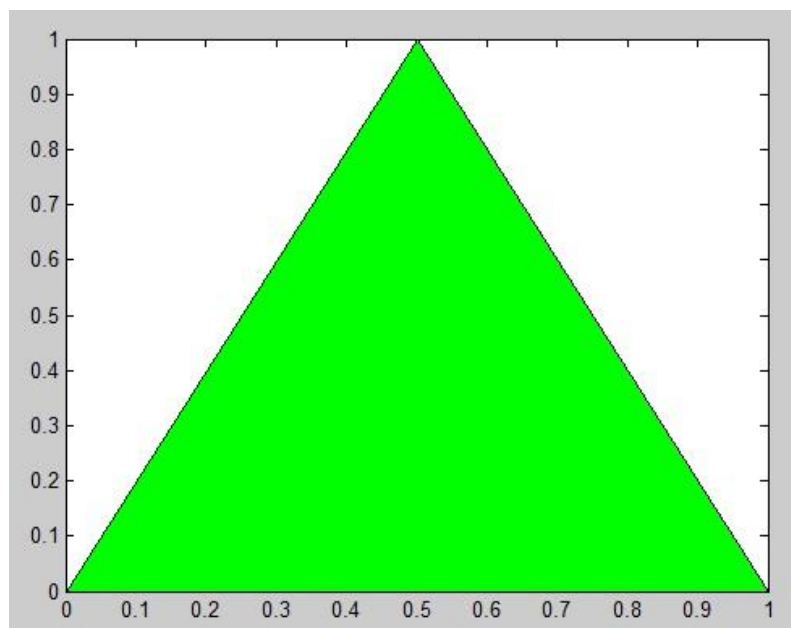
Základný graf funkcie  $\sin(x)$



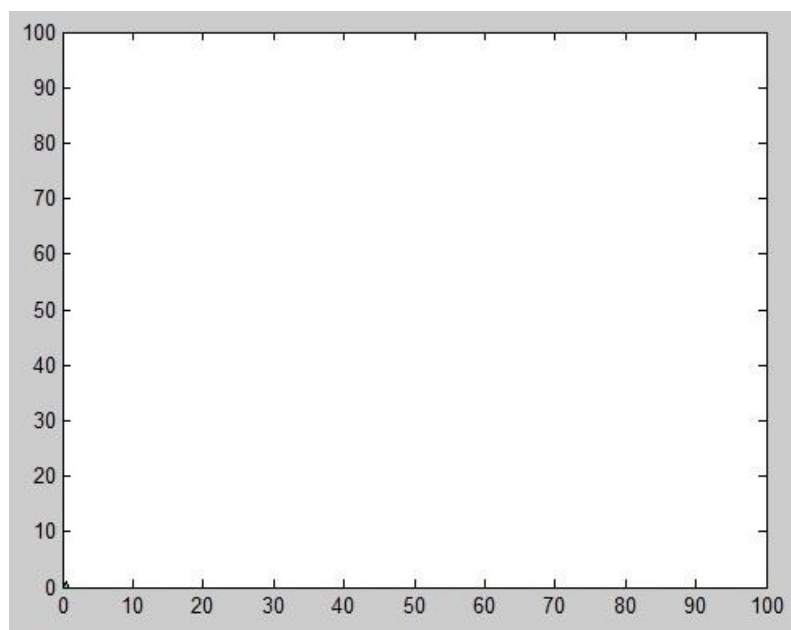
Spojenie grafov  $\sin(x)$  a  $\cos(x)$



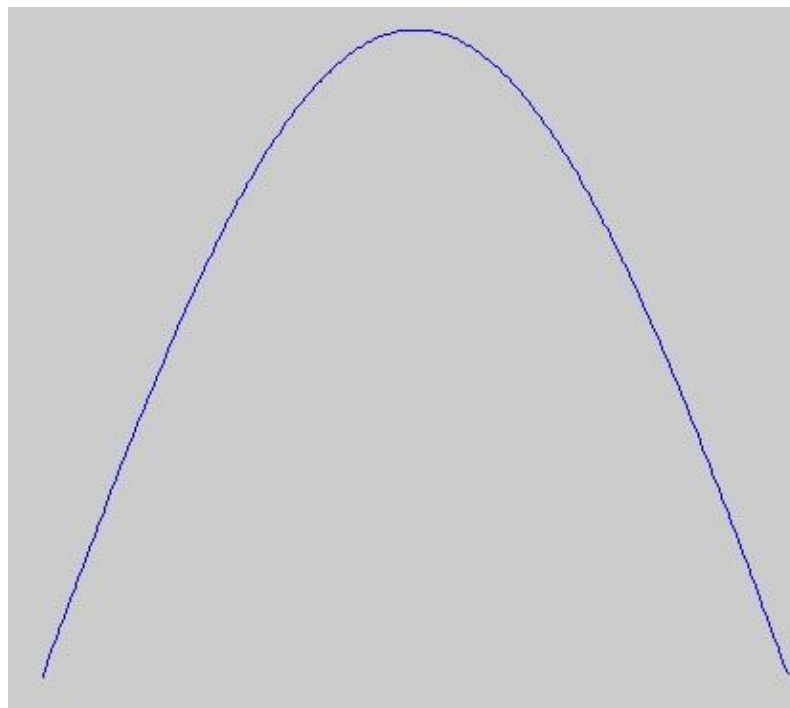
### Vyplnenie dvojrozmerného mnohouholníka



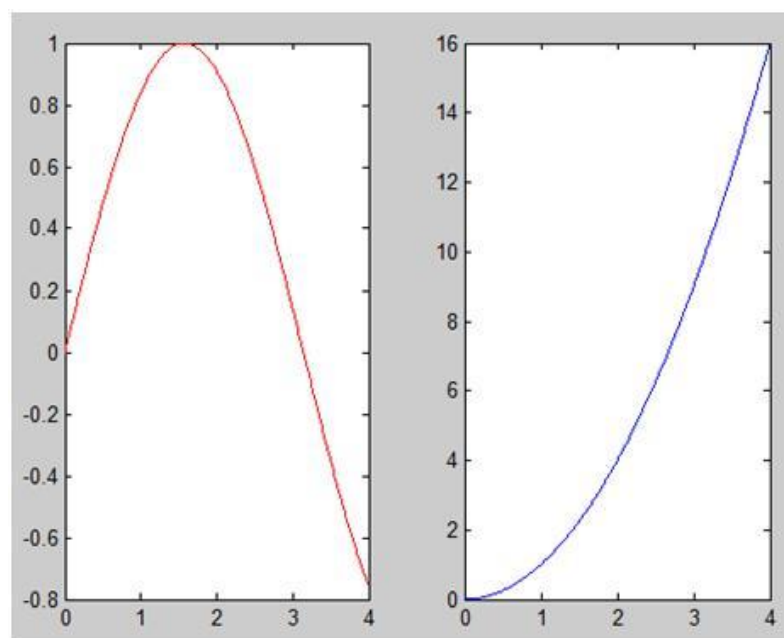
### Nastavenie rozmerov osí x, y



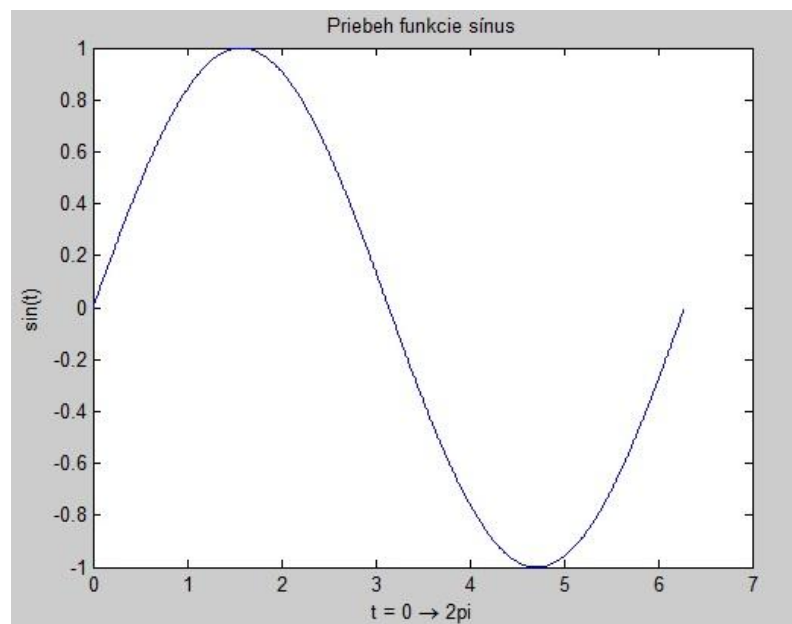
### Vypnutie zobrazenia osí



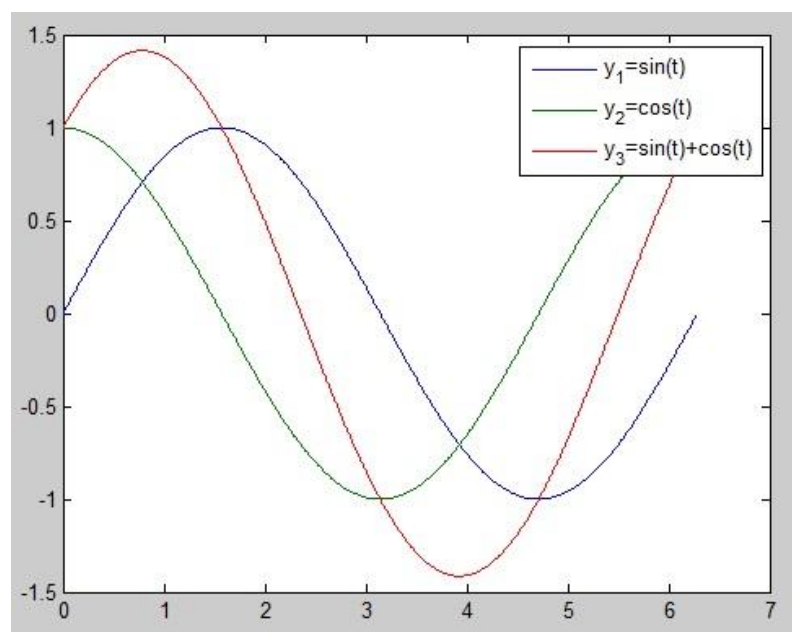
### Rozdelenie grafického okna na subokná



## Vytvorenie popisu grafu

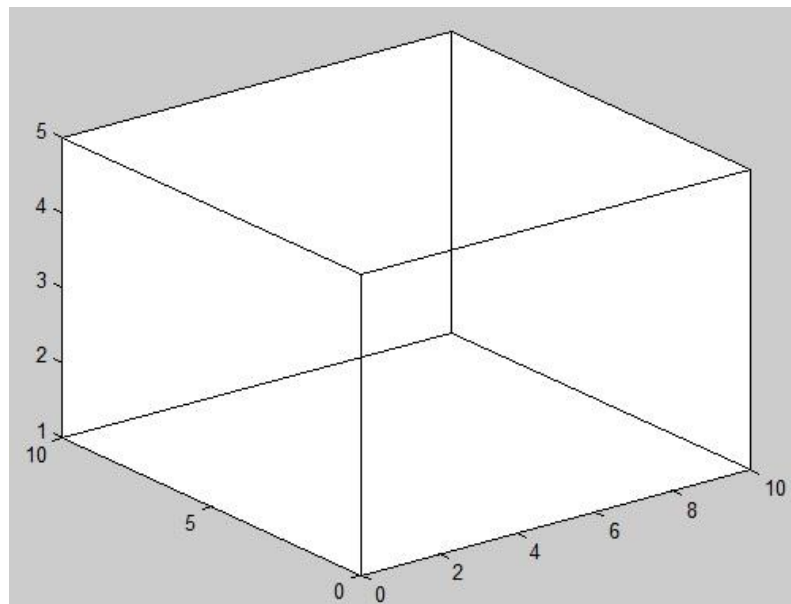


## Vytvorenie legendy grafu

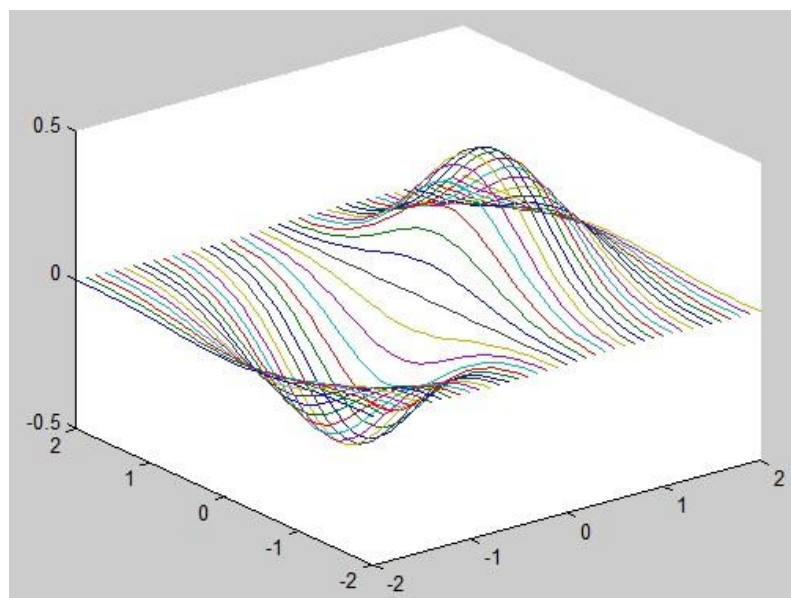


## PRÍLOHA P III: 3D GRAFY

Nastavenie rozmerov osí x, y a z



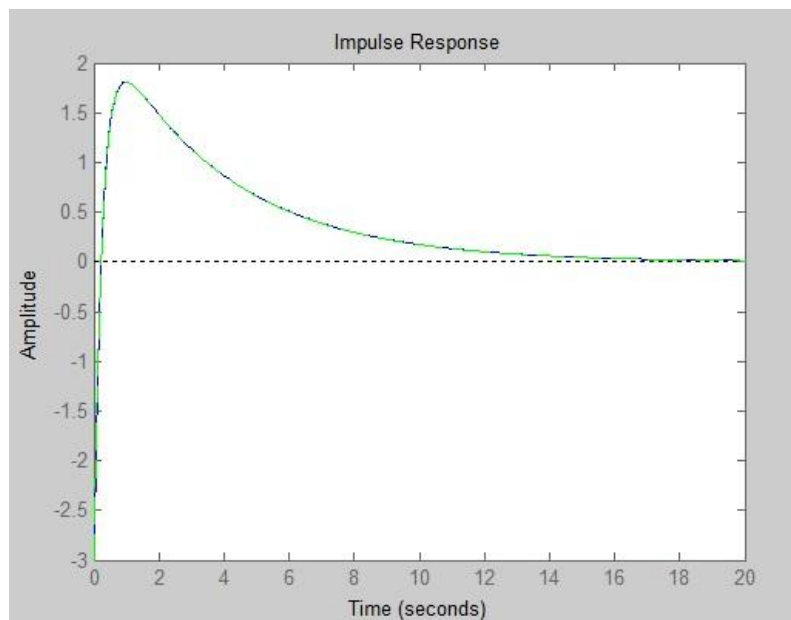
Vytvorenie trojrozmerného obrazca



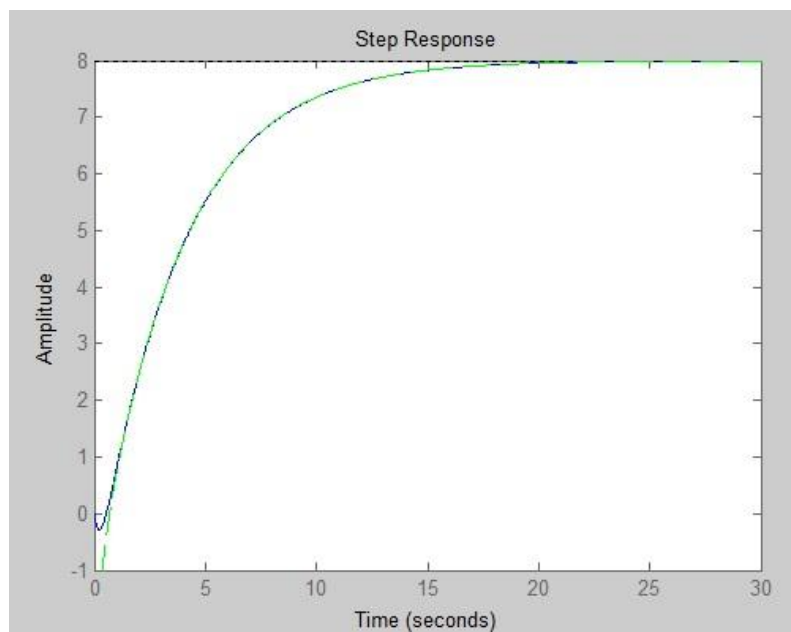


## PRÍLOHA P IV: NAJPOUŽÍVANEJŠIE GRAFY V OBLASTI AUTOMATIZÁCIE A TEÓRIE SYSTÉMU

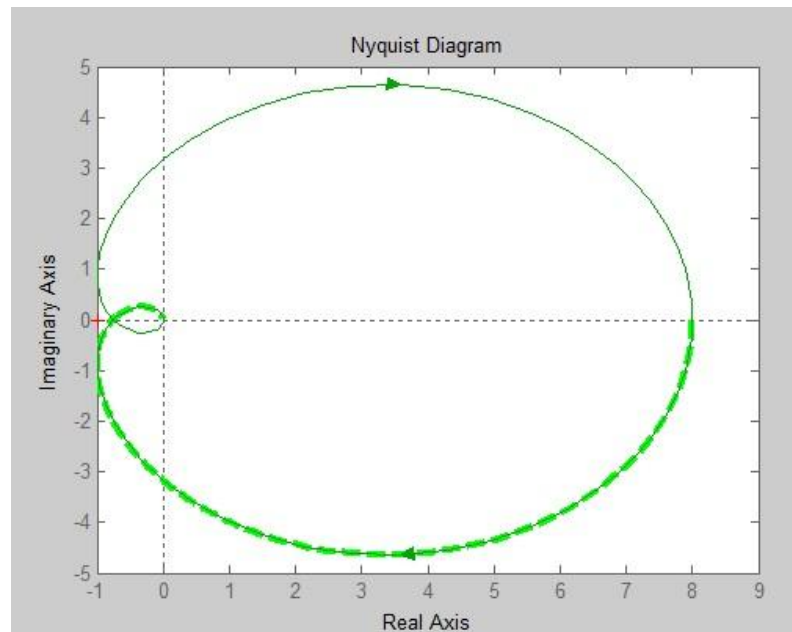
### Impulzová charakteristika



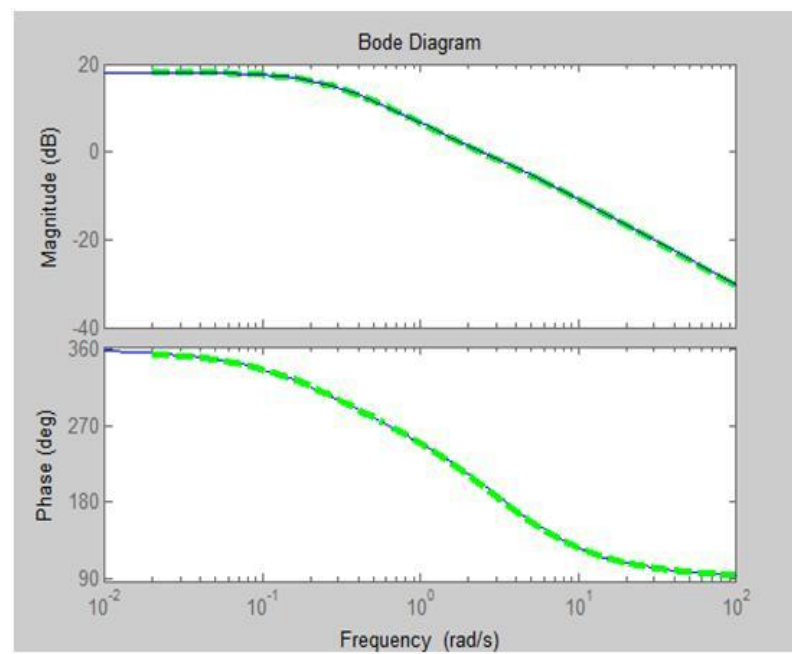
### Prechodová charakteristika



## Nyquistova krivka



## Bodeho krivka



## PRÍLOHA V: SCHÉMY ZO SIMULINKU

Schéma zobrazujúce rovnicu kružnice s polomerom  $r$

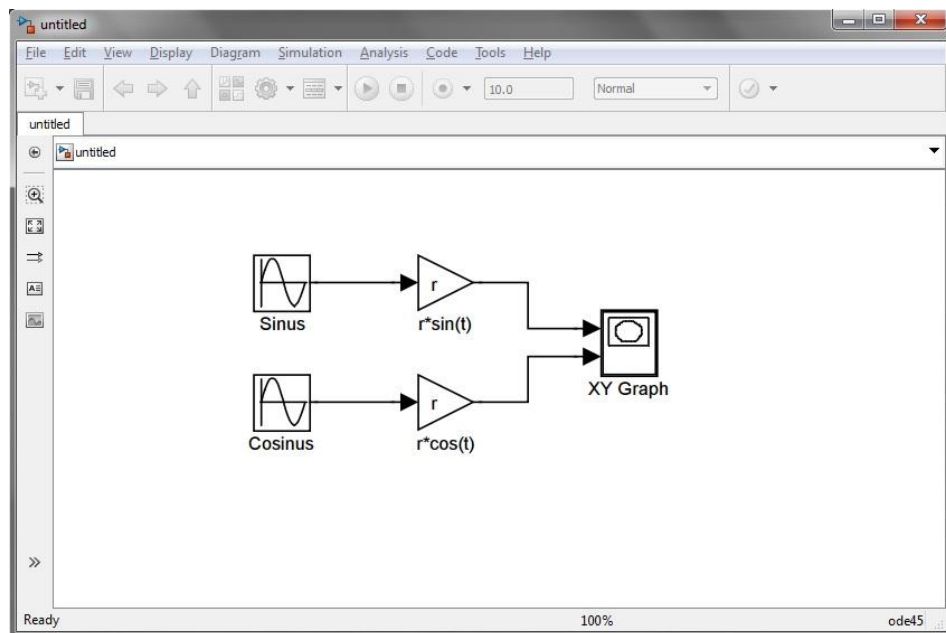
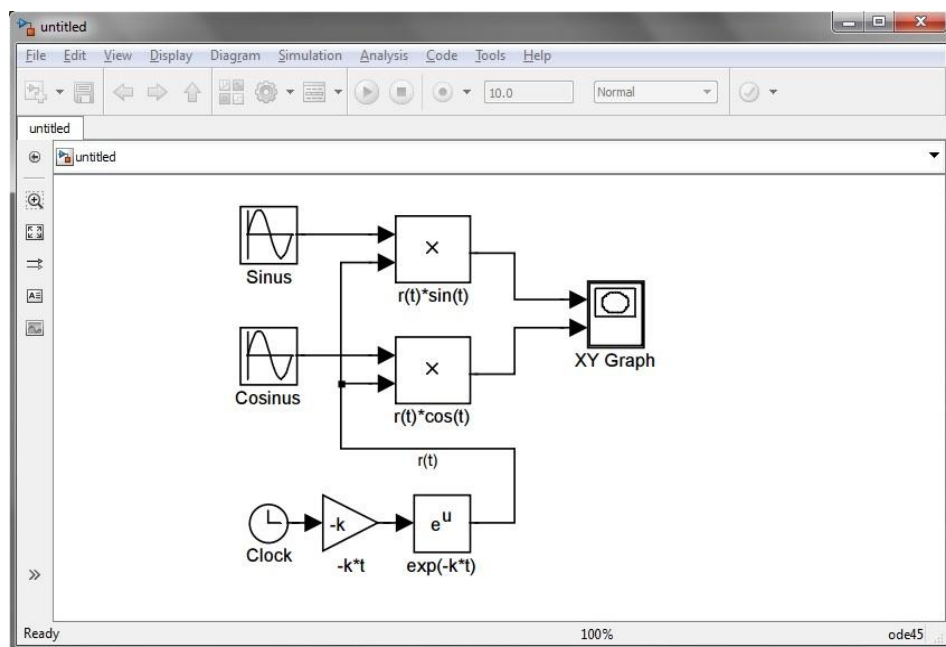
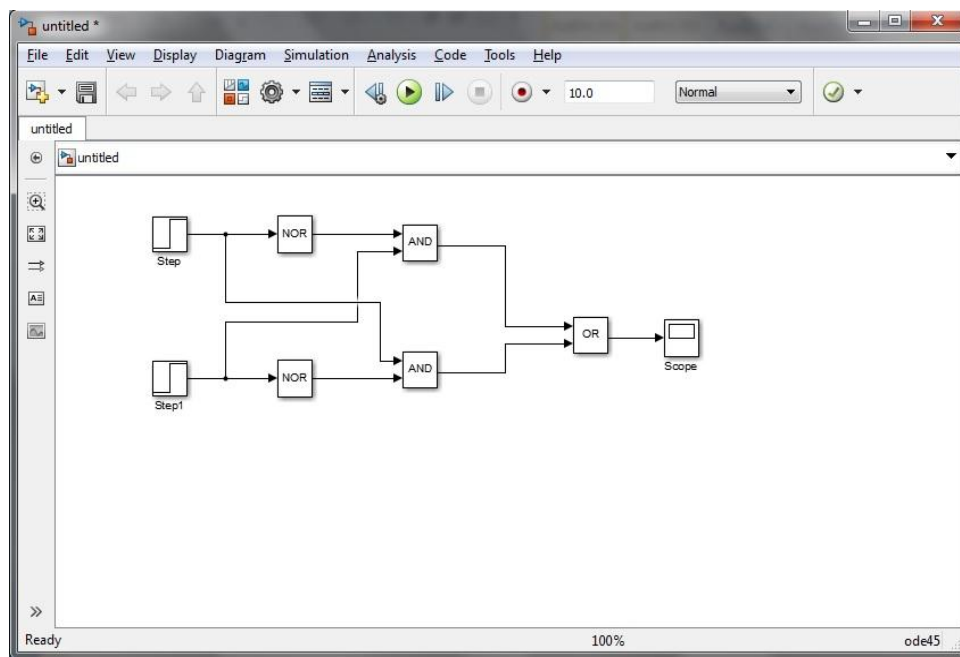


Schéma zobrazujúce model logaritmickéj špirály



## Schéma logickej funkcie



## Výstup PID regulátoru

