

Elektronická učební pomůcka pro předmět počítačová podpora automatického řízení

Michal Pítr

Bakalářská práce
2005



Univerzita Tomáše Bati ve Zlíně
Fakulta technologická

Vložit oficiální zadání bakalářské práce

Děkuji vedoucímu bakalářské diplomové práce panu Ing. Marku Kubalčíkovi za odbornou radu a vstřícnost v průběhu řešení práce.

Souhlasím s tím, že s výsledky mé práce může být naloženo podle uvážení vedoucího bakalářské práce a ředitele ústavu. V případě publikace budu uveden jako spoluautor.

Prohlašuji, že jsem na celé bakalářské práci pracoval samostatně a použitou literaturu jsem citoval.

Ve Zlíně, 08. 06. 2005

.....

podpis

ABSTRAKT

Abstrakt česky

Bakalářská práce byla vyhotovena jako učební pomůcka pro výuku předmětu počítačová podpora automatického řízení. Měla by pomoci studentům při výuce předmětu počítačová podpora automatického řízení, jehož náplní je práce v prostředí MATLAB - Simulink.

Práce obsahuje příklady elementárních matematických výpočtů, operací s vektory a maticemi, ukázky práce s textovými řetězci, tvorby grafů, využití prostředí Simulink pro oblast automatického řízení, tvorby a ukázky audia, videa i práci s obrázky.

Abstrakt ve světovém jazyce

This bachelor's thesis has been written to serve as teaching material for the Computer – Supported Automatic Control course. It should help students engaged in this course to acquire sufficient knowledge on working in MATLAB – Simulink environment.

My bachelor's thesis contains examples of basic mathematical operations, vector and matrix operations, samples of using text strings, plotting graphs, using the Simulink environment in automatic control, samples of making audio and video files as well as image processing.

OBSAH

ÚVOD.....	6
1 TEORETICKÁ ČÁST.....	9
1.1 Elementární matematika.....	10
1.1.1 Exponenciální funkce	10
1.1.2 Komplexní čísla.....	10
1.1.3 Zaokrouhlování a zbytek	11
1.1.4 Diskrétní matematika.....	11
1.1.5 Trigonometrické funkce	11
1.2 Operace s vektory a maticemi.....	12
1.2.1 Elementární matice	14
1.2.2 Speciální matice.....	15
1.2.3 Základní informace o poli a manipulace s maticemi	15
1.2.4 Operace s maticemi	16
1.3 Práce s textovými řetězci.....	17
1.4 Grafy.....	18
1.4.1 Základní grafy	19
1.4.2 Popisky grafů.....	22
1.4.3 Specializované grafy	24
1.5 Jednoduché úlohy pro MATLAB – Simulink	34
1.5.1 Vytvoření modelu a jeho spuštění v prostředí Simulink	35
1.5.2 Standardní knihovny prostředí Simulink.....	36
1.5.3 PI regulátor, PID regulátor, Smithův prediktor	36
1.6 Využití CONTROL TOOLBOXu	38
1.6.1 Přechodová charakteristika.....	39
1.6.2 Impulsní charakteristika	39
1.6.3 Amplitudově fázová frekvenční charakteristika	40
1.7 Audio, video a práce s obrázky.....	41
1.7.1 Obecné audio funkce	41
1.7.2 Funkce pro Microsoft WAVE	42
1.7.3 AVI funkce	43
1.7.4 Práce s obrázky.....	43
1.7.5 Obecná teorie o animaci	44
2 PRAKTICKÁ ČÁST.....	45
2.1 Panely nástrojů (roletové menu) programu	45
2.2 Hlavní programové okno	46
ZÁVĚR.....	47
SEZNAM POUŽITÉ LITERATURY	48
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	49
SEZNAM OBRÁZKŮ.....	50
SEZNAM TABULEK	51
SEZNAM PŘÍLOH.....	52

ÚVOD

V rychle se vyvíjejícím světě výpočetní techniky a programového vybavení mají od prvopočátku nezastupitelné místo programy pro numerické výpočty. Zpočátku byla podpora uživatele realizována matematickými knihovnami pro obecné programovací jazyky. Později se vyvinuly samostatné programy, které je možné rozdělit do tří oblastí - tabulkové procesory s rozšířenými matematickými funkcemi, balíky na řešení určitého okruhu problémů (statistika, mechanika) a profesionální univerzální balíky. Do této poslední oblasti patří mimo jiné i program MATLAB.

Co je to MATLAB? Název MATLAB vznikl z anglického *matrix laboratory*. MATLAB byl napsán, aby poskytoval jednoduchý přístup k matematickým knihovnám. Byl původně určen pro operační systém UNIX a tato okolnost se dodnes projevuje ve velmi jednoduchém základním komunikačním rozhraní – příkazové řádce.

MATLAB je vysoce výkonný jazyk pro technické výpočty s velkým množstvím funkcí, umožňujících snadné řešení technických problémů, speciálně takových, které vedou na vektorovou nebo maticovou formulaci v mnohem kratším čase, než řešení v klasických programovacích jazycích, jako je například program C++.

Typické oblasti použití jsou :

- inženýrské výpočty
- vývoj algoritmů
- modelování a simulace při vývoji prototypů
- analýza dat a jejich vizualizace
- inženýrská grafika
- vývoj aplikací, včetně tvorby grafického uživatelského rozhraní

V univerzitním prostředí jde o standardní nástroj využívaný ve výuce matematiky a v inženýrských oborech. V průmyslu je využíván jako vysoce efektivní nástroj pro výzkum, analýzu a vývoj dat.

Tato bakalářská práce a program, který je její součástí, byly napsány jako materiál, sloužící pro výuku programu MATLAB a prostředí Simulink, což je grafická nádstavba programu MATLAB, ve které je možno pomocí jednotlivých bloků sestavovat různá zapojení a modely.

V této práci je objasněna nejčastější problematika, která nastává při práci s programem MATLAB, a to včetně demonstračních úloh a příkladů.

Je třeba hned na začátku upozornit na předpoklad, že čtenář, či uživatel má možnost program MATLAB používat, příklady si odzkoušet a využívat nápovědu. Část této práce je zaměřena na téma automatizace a souvisejících oblastí.

„Elektronická učební pomůcka pro předmět počítačová podpora automatického řízení“, což je vlastně název celé Bakalářské práce, se dělí na několik základních kapitol.

První část zvaná „Elementární matematika“ je zaměřena na jednodušší výpočty. Tato sekce je rozdělena do pěti dalších kapitol, z nich každá má svou teoretickou i praktickou část. Tedy každá teoretická část nás seznámí s tvorbou jednotlivých příkazů, kdežto v části praktické můžeme přímo odzkoušet své teoretické znalosti. Tato kapitola tedy obsahuje exponenciální funkce, komplexní čísla, zaokrouhlování a zbytek, diskrétní matematiku a trigonometrické funkce.

Druhá kapitola má název „Operace s vektory a maticemi“. Jednotlivé části této kapitoly nás seznámí s tvorbou elementárních a speciálních matic, dále obsahuje základní informace o poli a manipulace s maticemi a poslední část této kapitoly je operace s maticemi, tedy jejich součet, rozdíl, násobení, dělení atd. Tato kapitola je důležitou částí bakalářské práce, protože matice a jejich operace jsou v programu MATLAB podstatnou částí, na kterých je postavený celý program MATLAB.

Třetí kapitola nese název „Operace s textovými řetězci“. Po zadání dvou textových řetězců můžeme provádět různé operace, jako jsou například: porovnání řetězců, srovnávání prvních n znaků, zarovnávání řetězců, nahrazování řetězců atd.

Čtvrtá kapitola má název „Grafy“. Cílem této kapitoly je seznámit uživatele s celou řadou grafů, které nám program MATLAB nabízí. Po absolvování této části by měl být uživatel schopen sestavit libovolný až 2D nebo 3D graf včetně jeho popisu.

Pátá kapitola programu má název „MATLAB - Simulink“. Simulink je prostředí, které slouží k simulaci dynamických systémů s využitím grafických bloků. Toho je v programu využito k jednoduché demonstraci elementárních úloh z teorie automatického řízení. Kapitola je rozdělena na části: „Uzavřený regulační obvod s PID regulátorem“, „Uzavřený regulační obvod s PI regulátorem“, „Regulační obvod s využitím Smithova prediktoru“ a „Nejpoužívanější bloky v teorii automatického řízení“. Uživatel si může v každé z těchto částí nadefinovat vlastní typ přenosu a nastavit regulátor podle vypočtených parametrů. Vše je vytvořeno v prostředí Simulink, což umožňuje velmi jednoduchou grafickou obsluhu jednotlivých bloků při zadávání vlastních hodnot.

Předposlední kapitola má název „Využití CONTROL TOOLBOXu“. Slouží pro zjednodušení a ulehčení práce při počítání průběhů jednotlivých charakteristik (Přechodová charakteristika, Impulsová charakteristika, Amplitudově fázová frekvenční charakteristika a Logaritmická amplitudově fázová frekvenční charakteristika) při zadaném přenosu $G(s)$. Uživatel si definuje svůj vlastní přenos, poté má možnost si programem nechat vypočítat jednotlivé body a zobrazit danou charakteristiku. Kapitola umožňuje získat více informací o zobrazení charakteristik a uživatel má možnost velmi rychle získat představu o tvaru a průběhu charakteristik jednotlivých systémů.

V poslední kapitole, která má název : „Audio,video a práce s obrázky“ je třeba ukázat možnosti programu MATLAB. Kapitola je rozdělena do čtyř sekcí a každá z těchto sekcí obsahuje jeden příklad a teorii. Příklady demonstrují a popisují zajímavé funkce v programu MATLAB jak pro tvorbu audio,video, tak i obrázků..

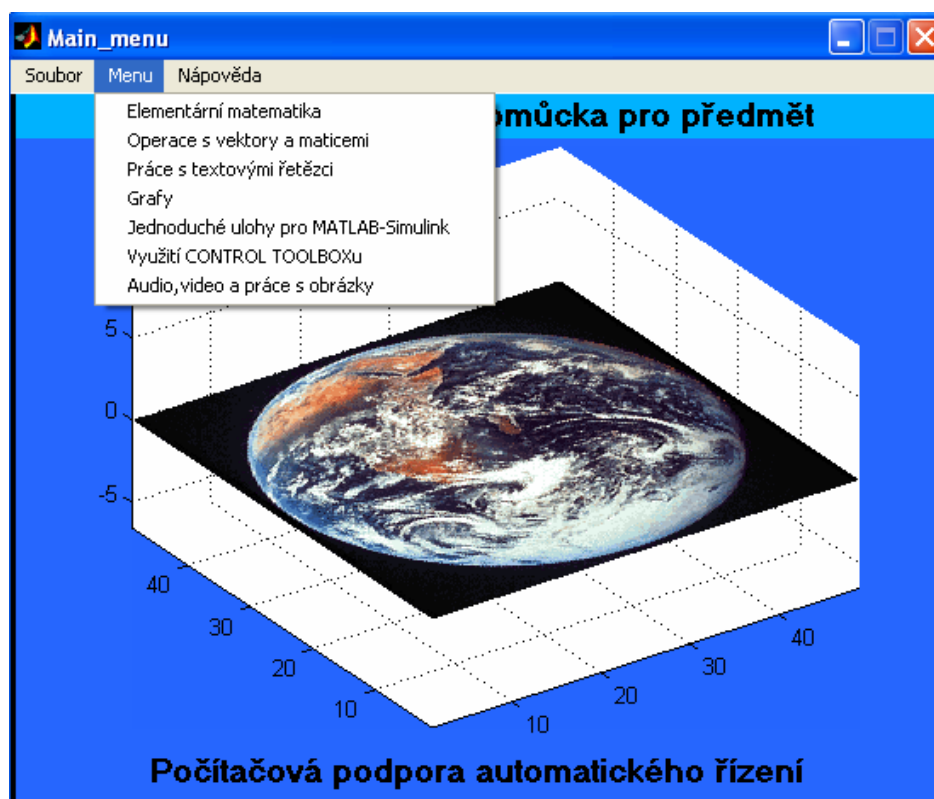
Elektronická učební pomůcka pro předmět počítačová podpora automatického řízení se spouští příkazem „main“ v prostředí MATLABu.

1 TEORETICKÁ ČÁST

Teoretická část by měla dát základní informace o používání elektronické učební pomůcky. Dále poskytuje přehled všech použitých funkcí v programu. Na začátku je třeba zdůraznit aspoň dvě skutečnosti, které je vhodné si uvědomit. MATLAB používá poměrně hutný způsob zápisu příkazů, což je na jednu stranu příjemné, neboť toho nemusíme mnoho napsat, na druhou stranu tento zápis není příliš přehledný. Je důležité si uvědomit, že každý znak má své místo a svůj význam. Je třeba dodržovat tzv. syntaxi jazyka, tedy dodržovat věci, které jsou jasné lidem, znalým programovacích jazyků, ale nejasné lidem, kteří programovací jazyk nepoužívají.

Druhým, zdánlivě samozřejmým faktem, je přečíst si důkladně informaci, která se vypíše při provádění příkazu a zjištění nějaké chyby.

Podle menu programu „Elektronická učební pomůcka pro předmět počítačová podpora automatického řízení“ (viz. Obr.1) je tato teoretická část rozdělena do několika oblastí.



Obr. 1 Menu programu „Elektronická učební pomůcka pro předmět počítačová podpora automatického řízení“

1.1 Elementární matematika

Tato oblast se zabývá jednoduchými matematickými výpočty exponenciálních funkcí, komplexních čísel, zaokrouhlování a zbytku, diskrétní matematiky a trigonometrických funkcí. Popis teorie jednotlivých matematických funkcí přesahuje rozsah této práce, tato kapitola se spíše věnuje jednotlivým příkazům.

1.1.1 Exponenciální funkce

Tab. 1 Příkazy pro práci s exponenciálními funkcemi

<i>n</i>	Příkazy	Popis jednotlivých příkazů
1.	exp(x)	- e na argument x
2.	log(x)	- přirozený logaritmus
3.	log2(x)	- binární logaritmus
4.	log10(x)	- dekadický logaritmus
5.	nextpow2(x)	- nejbližší mocnina 2
6.	pow2(x)	- mocnina 2 na x
7.	reallog([...])	- přirozený logaritmus nezáporného reálného pole
8.	sqrt(x)	- druhá odmocnina

n – číslo příkazu

1.1.2 Komplexní čísla

Tab. 2 Příkazy pro práci s komplexními čísly

<i>n</i>	Příkazy	Popis jednotlivých příkazů
1.	abs(x)	- absolutní hodnota komplexního čísla z
2.	angle(z)	- fázový úhel komplexního čísla z
3.	complex(a,b)	- vytvoří komplexní Číslo z reálné(a) a imaginární(b) složky
4.	imag(z)	- imaginární část komplexního čísla z
5.	real(z)	- reálná část komplexního čísla z

n – číslo příkazu

1.1.3 Zaokrouhlování a zbytek

Tab. 3 Příkazy pro práci se zaokrouhlováním a se zbytkem

<i>n</i>	Příkazy	Popis jednotlivých příkazů
1.	fix(x)	- zaokrouhlení k nule
2.	floor(x)	- zaokrouhlení k minus nekonečnu
3.	ceil(x)	- zaokrouhlení k plus nekonečnu
4.	round(x)	- zaokrouhlení k nejbližšímu celému číslu
5.	mod(a,b)	- modul po dělení, kde a je číselník a b je jmenovatel
6.	rem(a,b)	- zbytek po dělení, kde a je číselník a b je jmenovatel
7.	sign(x)	- funkce signum

n – číslo příkazu

1.1.4 Diskrétní matematika

Tab. 4 Příkazy pro diskrétní matematiku

<i>n</i>	Příkazy	Popis jednotlivých příkazů
1.	factor(x)	- nejmenší a největší dělitel x
2.	factorial(x)	- faktoriál čísla x
3.	gcd(a,b)	- největší společný dělitel čísel a,b
4.	lcm(a,b)	- nejmenší společný násobek
5.	isprime(x)	- test, je-li číslo x prvočíslo
6.	perms[...]	- vytvoří v matici všechny možné permutace

n – číslo příkazu

1.1.5 Trigonometrické funkce

Tab. 5 Příkazy pro práci s trigonometrickými funkcemi

<i>n</i>	Příkazy	Popis jednotlivých příkazů
1.	Acos(x)	- arkuskosinus
2.	acosh(x)	- hyperbolický arkuskosinus
3.	acot(x)	- arkuskotangens
4.	acoth(x)	- hyperbolický arkuskotangens
5.	acsc(x)	- arkuskosekanus
6.	acsch(x)	- hyperbolický arkuskosekanus
7.	asec(x)	- arkussekans
8.	asech(x)	- hyperbolický arkussekans
9.	asin(x)	- arkussinus
10.	sinh(x)	- hyperbolický arkussinus
11.	atan(x)	- arkustangens
12.	atanh(x)	- hyperbolický arkustangens

13.	$\cos(x)$	- kosinus
14.	$\cosh(x)$	- hyperbolický kosinus
15.	$\cot(x)$	- kotangens
16.	$\coth(x)$	- hyperbolický kotangens
17.	$\csc(x)$	- kosekans
18.	$\operatorname{csch}(x)$	- hyperbolický kosekans
19.	$\sec(x)$	- sekans
20.	$\operatorname{sech}(x)$	- hyperbolický sekans
21.	$\sin(x)$	- sinus
22.	$\sinh(x)$	- hyperbolický sinus
23.	$\tan(x)$	- tangens
24.	$\tanh(x)$	- hyperbolický tangens

n – číslo příkazu

1.2 Operace s vektory a maticemi

Tato oblast se zabývá základními operacemi s maticemi a vektory, jejich vytváření a plnění hodnotami, tvorbou elementárních i speciálních matic a také jejich manipulacemi. Je zde objasněn postup vytvoření některých speciálních matic, práce se sloupcem či řádkem matice, dále demonstrování rozdílu mezi maticovými operacemi, operacemi s jednotlivými prvky matice a také je zde popsána tematika elementárních funkcí pro násobení, sčítání, odčítání, dělení a výpočet vlastních čísel matice.

Pro **tvorbu matic** platí určitá pravidla. Skalár je matice s oběma rozměry rovnými jedné. Vektor je matice, kdy jeden rozměr má hodnotu rovnající se jedné. Rozlišujeme vektory sloupcové a řádkové. Je zvykem, že první rozměr označuje počet řádků a druhý počet sloupců. Řekneme-li, že matice je rozměru 7x12, to znamená že má 7 řádků a 12 sloupců. Toto platí jak v matematických zápisech, tak i v programu MATLAB.

Pro vyjádření opačného vektoru se používá symbol transpozice který určuje, že matice 5x2 se stane maticí 2x5. V MATLABu používáme za znak transpozice $'$ (apostrof).

Jazyk MATLAB neobsahuje žádný příkaz pro nastavení dimenze nebo typu matice. Potřebnou paměť alokuje automaticky až do velikosti využitelné na konkrétním počítači.

Nejjednodušším způsobem zadávání matic je napsat matici jako seznam prvků, za dodržení následujících pravidel:

- jednotlivé prvky matice v řádce oddělit mezerou nebo čárkou
- jednotlivé řádky matice oddělit středníkem nebo znakem konce řádky
- celý seznam vložit mezi hranaté závorky

Pro **plnění vektorů a matic** platí taktéž určitá pravidla. Zápis vektorů a, b, c a matice A v MATLABu pak vypadá následovně:

$A=[1;2;3;4;5]$

Sloupcový vektor (oddělovač středník) s výpisem (na konci příkazu není středník)

$A=$ 1
 2 *pozn.: Takto vypadá výpis proměnné A v programu MATLAB*
 3
 4
 5

$B=[1,2,3,4,5]$

Řádkový vektor (oddělovač čárka nebo mezera) s výpisem

$B=$ 1 2 3 4 5

Zápis matice s uvedením všech prvků :

$A=[1,2,3,4;5,6,7,8]$

$A=$ 1 2 3 4
 5 6 7 8

Využijeme – li existujícího vektoru B , můžeme matici definovat jednodušeji:

$A=[B,B]$

$A=$ 1 2 3 4 5
 1 2 3 4 5

Pro **práci s částí matice nebo vektoru** platí také pravidla. V těchto příkladech využijeme již dříve nadefinované matice. Práce s pouze jednou hodnotou vektoru či matice je standardní.

$A(2)$ zjištění hodnoty druhého prvku vektoru A ,

výsledek: $ans = 2$

$B(5) = x$ (2,3) hodnotu z druhého řádku a třetího sloupce matice „ x “ dej do pátého prvku vektoru C ,

výsledek: $C=$ 1 2 3 4 5 7

V MATLABu lze také pracovat s částí vektoru či submaticí matice. V tomto případě se používá pro určení rozsahu indexů znak dvojtečka. Např. část matice „ x “ (submatici) lze získat příkazem:

```
x(2:5,1:2) ans=      3      32
                   31      9
                   8      28
                   30      5
```

Často je potřebné vyjmout celý sloupec nebo řádek. To je přirozeně možné i předchozím způsobem. Pomocí příkazu „**size**“ zjistíme rozměry a tyto poté použijeme pro určení rozsahu. Existuje snadnější způsob, a to použitím samotné dvojtečky na místě příslušného indexu získáme celý rozsah.

```
y=x(3,:)    pozn.: vyjmi třetí řádek, ulož do proměnné „y“ a vypiš
y= 31  9  2  22  27  20
```

1.2.1 Elementární matice

V MATLABu často potřebujeme užívat matice a vektory naplněné speciálními hodnotami. Jedním z nejčastěji se vyskytujících požadavků je vytvoření matice zadaných rozměrů, naplněných konstantou. MATLAB poskytuje dvě funkce pro tyto účely. Funkce „**zeros**“ vytvoří a naplní matici hodnotou nula a funkce „**ones**“ hodnotou jedna. Následující seznam uvádí použité příklady funkcí vytvářejících elementární matice:

<code>x=zeros(11,50)</code>	matice o rozměrech 11x50 naplněná nulami
<code>x=zeros(100)</code>	nulová čtvercová matice 100x100
<code>x=ones(1,1000)</code>	řádkový vektor o 1000 prvcích plný hodnot 1
<code>x=eye(100)</code>	jednotková matice o rozměru 100x100
<code>x=eye(25,50)</code>	matice 25x50 s jednotkami na hlavní diagonále
<code>x=rand(5,7)</code>	matice 5x7 naplněná náhodnými čísly v intervalu (0,1)

Tab. 6 Příkazy pro tvorbu elementárních matic

<i>n</i>	Příkazy	Popis jednotlivých příkazů
1.	zeros(m,n)	- nulová matice, kde m je počet řádků a n počet sloupců
2.	ones(m,n)	- jedničková matice, kde m je počet řádků a n počet sloupců
3.	eye(m,n)	- jednotková matice, kde m je počet řádků a n počet sloupců
4.	rand(m,n)	- stejněměrná náhodně distribuovaná matice, kde m je počet řádků a n je počet sloupců
5.	randn(m,n)	- normální náhodně distribuovaná matice, kde m je počet řádků a n je počet sloupců
6.	linspace(k,l,n)	- lin.rozdělení matice na n bodů, kde k je horní mez a l je dolní mez
7.	logspace(k,l,n)	- log.rozdělení matice na n bodů, kde k je horní mez a l je dolní mez
8.	freqspace(n)	- stanovení frekvenčního intervalu pro frekvenční odezvu

n – číslo příkazu

1.2.2 Speciální matice

Tab. 7 Příkazy pro tvorbu speciálních matic

<i>n</i>	Příkazy	Popis jednotlivých příkazů
1.	compan(c)	- doprovodná matice, kde c je vektor koeficientu polynomu
2.	hadamart(c)	- Hadamartova matice, kde c je jakákoliv mocnina dvou
3.	hankel(c)	- Hankelova matice, kde c je první sloupec a její prvky jsou nulové pod první diagonálou
4.	hilb(c)	- Hilbertova matice, kde c je jakékoliv číslo
5.	invhilb(c)	- inverzní Hilbertova matice
6.	magis(c)	- Magický čtverec – jde o čtvercovou matici, kde součet prvků v každém sloupci i řádku je stejný
7.	pascal(c)	- Pascalova matice – symetricky pozitivně definitní matice celých čísel z Pascalova trojúhelníku
8.	toeplitz(c)	- Toeplitzova matice, kde c je libovolný vektor
9.	vander(c)	- Vanermondeova matice, kde c je libovolný vektor
10.	wilkinson(w)	- Wilkinsonova matice, kde c je libovolný vektor

n – číslo příkazu

1.2.3 Základní informace o poli a manipulace s maticemi

Tab. 8 Příkazy pro práci s informacemi a manipulacemi matic

<i>n</i>	Příkazy	Popis jednotlivých příkazů
1.	size(a)	- číselná hodnota rozměru pole(matice)
2.	length(a)	- číselná hodnota délky vektoru
3.	ndims(a)	- číselná hodnota počtu dimenzí v poli
4.	numel(a)	- číselná hodnota počtu prvků pole

5.	disp(a)	- zobrazení pole
6.	isempty(a)	- test, je-li pole prázdné (ano-1, ne-0)
7.	isequal(a,b)	- test, jsou-li matice A a B stejného typu, velikosti a obsahu
8.	cat(1,a,b)	- sloučení polí (matic)
9.	reshape(a,m,n)	- změna rozměrů pole, kde m je počet řádků a n sloupců nové mat.
10.	diag(a,k)	- diagonální matice, kde k je diagonála (k=0 je hlavní diagonála)
11.	blkdiag(a,b)	- bloková diagonální matice
12.	tril(a)	- dolní trojúhelníková část matice
13.	triu(a)	- horní trojúhelníková část matice
14.	fliplr(a)	- levo pravá rotace
15.	flipud(a)	- horně dolní rotace
16.	flipdim(a,dim)	- rotace podle dimenze
17.	rot90(a)	- rotace o 90°
18.	find(doa)	- indexy nenulových prvků matice
19.	ind2sub(s,IND)	- vektorové indexy z lineárních indexů
20.	sub2ind(s,IND)	- lineární indexy z vektorových indexů

n – číslo příkazu

1.2.4 Operace s maticemi

V případě, že budeme potřebovat získat hodnoty z násobení, dělení, sčítání nebo odčítání matic, použijeme pro to následující elementární operace:

Nejprve si nadefinujeme matice A a B , dále můžeme podle přiložené tabulky č.9 vypočítat jednotlivé operace.

Tab. 9 Příkazy pro operaci s maticemi

Matice	Matice	Operátor	Název	Zápis v MATLABu
A	B	+	Součet	$X=A+B$
A	B	-	Rozdíl	$X=A-B$
A	B	*	Násobení	$X=A*B$
A	B	/	Dělení levé	$X=A/B$
A	B	\	Dělení pravé	$X=B\backslash A$
A	B	^	Umocňování	$X=A^B$
A	B	.*	Násobení prvků matic	$X=A.*B$
A	B	./	Dělení levé prvků matic	$X=A./B$
A	B	.\	Dělení pravé prvků matic	$X=B.\backslash A$
A	B	.^	Umocňování prvků matic	$X=A.^B$
A		diag	Hodnoty na hlavní diagonále	$X=\text{diag}(A)$
A		eig	Vlastní čísla matice	$X=\text{eig}(A)$

1.3 Práce s textovými řetězci

V této kapitole se budeme věnovat příkazům pro práci s řetězci.

Převod řetězce na číslo:

str2double('řetězec') – na double, pro čtení čísel z dialogových oken
 str2num – na numer(výchozí formát čísla), pro čtení čísel z dialogových oken
 bin2dec('00101') – řetězec binárního čísla na desítkové číslo
 hex2dec('3ff') – řetězec hexadecimálního čísla na desítkové číslo
 hex2num – hexadecimální na double

Převod čísla na řetězec:

char – na pole znaků (do 127 ASCII tabulka)
 dec2bin – dekadické číslo na řetězec binárního čísla
 dec2hex – dekadické číslo na řetězec hexadecimálního čísla
 int2str – převede celé číslo na řetězec
 mat2str – převede matici na řetězec
 num2str – převede číslo na řetězec

Znaky a řetězce

Tab. 10 Příkazy pro práci se znaky a řetězci

<i>n</i>	Příkazy	Popis jednotlivých příkazů
1.	blanks	- řetězec mezer
2.	lower	- převede řetězec na malá písmena
3.	deblank	- zruší mezery na konci řetězce
4.	sprintf	- zápis formátovaných dat do řetězce používající pravidel C jazyka
5.	sscanf	- načtení řetězce podle formátu – pravidla C jazyka
6.	strjust	- srovnání textového řetězce
7.	strread	- načtení formátovaných dat z řetězce
8.	strep	- nahrazení části řetězce jiným
9.	upper	- převede řetězec na kapitálky
10.	findstr	- najde zadanou část řetězce, pokud není, tak zobrazí []
11.	regexp	- porovnání řetězců, např. hledání velkých písmen
12.	regexprep	- nahrazení části řetězce jiným podle kritérií
13.	strfind	- najde jeden řetězec ve druhém
14.	eval	- pokud je v řetězci nějaká funkce Matlabu, tak se provede a vypíše výsledek
15.	evalc	- to samé co eval, ale výsledek lze uložit do proměnné

n – číslo příkazu

1.4 Grafy

Grafický systém v MATLABu obsahuje různé způsoby pro zobrazení dat. Tento systém je vybudován na základě sady grafických objektů (*line*, *surface*, *atd....*), jejichž vzhled lze řídit nastavením parametrů jejich vlastností.

Protože MATLAB obsahuje dostatečné množství 2-D a 3-D grafických funkcí vyšší úrovně, není nezbytné používat přímý přístup k vlastnostem objektů.

V této kapitole je objasněn postup 2-D a 3-D grafických funkcí, jejich použití a tvorba jednotlivých popisků grafu.

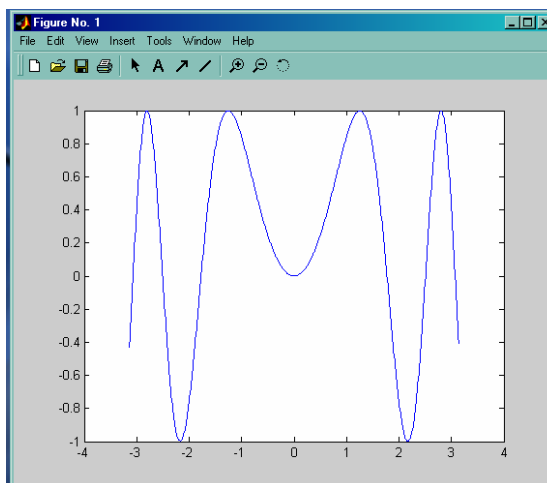
Základním příkazem pro vykreslení dvourozměrného grafu je příkaz „**plot**“. Pro úplný popis možností příkazu „plot“ můžeme použít nápovědu „help plot“.

Příklady funkcí a jejich průběhy:

Chceme – li například nakreslit funkci $y = \sin(x^2)$ na intervalu $x \in [-\pi, \pi]$ je třeba použít zápis v MATLABu:

```
x=-pi:0.01:pi;  
y=sin(x.^2);  
Plot(x,y)
```

Tím docílíme následujícího grafu:



graf 1. Ukázka grafu 2-D pro určenou funkci

Chceme - li nakreslit několik průběhů do jednoho grafu, musíme si připravit odpovídající vektory jednotlivých grafů. Užitečná funkce pro toto kreslení několika průběhů je „**fplot**“. Tato funkce je určená přímo na vykreslení průběhu funkce v zadaném rozsahu osy „x“ a případně i osy „y“.

1.4.1 Základní grafy

Cílem základních grafů je seznámit uživatele s:

- a) Vykreslením grafu s chybovými úsečkami
- b) Upřesněním stylu vykreslované křivky
- c) Vykreslením grafu v logaritmicko-logaritmických souřadnicích
- d) Vykreslením grafu v polárních souřadnicích
- e) Vykreslením vektorů nebo matic
- f) Vykreslením čar a bodů v 3D prostoru
- g) Vykreslením grafu s osou y vlevo i vpravo
- h) Umístěním více grafů vedle sebe do jednoho okna

a) Vykreslení grafu s chybovými úsečkami

příkaz **errorbar**(Y,E)

kde Y je vektor vykreslovaných dat

kde E je vektor chybových úseček

příkaz **errorbar**(X,Y,L,U)'

kde X versus Y

kde L je vektor dolních chybových úseček

kde U je vektor horních chybových úseček

Příklad pro vykreslení grafu:

```
x=0:pi/10:pi;
```

```
y=sin(x);
```

```
e=std(y)*ones(size(x));
```

```
errorbar(x,y,e)
```

b) Upřesnění stylu vykreslované křivky

LineStyle - syntaxe specifikace křivky při vykreslování

A) Specifikátory stylu křivky

specifikátor *styl křivky*

-	plná
--	čárkovaná
.	tečkovaná
-.	čerchovaná

B) Specifikátory označení bodu

specifikátor *styl křivky*

+	znak plus
o	znak kolečko
*	znak hvězdička
.	znak bodu
x	znak křížku
s	znak čtverce
d	znak kosočtverce

>	pravobodový trojúhelník
<	levobodový trojúhelník
p	pentagram(pěticípá hvězda)
h	hexagram(šesticípá hvězda)

C) Specifikátory barvy

specifikátor barva křivky

r	červená
g	zelená
b	modrá
c	světle modrá
m	fialová
y	žlutá
k	černá
w	bílá

hold - příkaz pro podržení aktuálního grafu tak, aby nebyl vymazán jiným

hold on - nové graf. objekty jsou přidány do grafu

hold off - nové graf. objekty nahradí vše předchozím grafu

hold - přepne na druhý stav (bud on nebo off)

c) Vykreslení grafu v logaritmicko-logaritmických souřadnicích

loglog - syntaxe pro vykreslení grafu v logaritmicko-logaritmických souřadnicích

Příklad pro vykreslení grafu:

```
x=logspace(-1,2);
loglog(x,exp(x),-s)
grid on
```

grid on - zapne v grafu mřížku

d) Vykreslení grafu v polárních souřadnicích

polar - syntaxe pro vykreslení grafu v polárních souřadnicích

polar(theta,rho)

theta - uhel'

rho - poloměr

polar(theta,rho,LineStyle)

LineStyle - specifikace křivky

Příklad pro vykreslení grafu:

```
t=0:.01:2*pi;
polar(t,sin(2*t).*cos(2*t),-r)
```

e) Vykreslení vektorů nebo matic

plot - syntaxe pro vykreslení vektorů nebo matic

plot(Y)

Y - vektor na Y ose, prvky mají na X ose hodnotu 1,2,3...

plot(X1,Y1,X2,Y2,...)
více veličin do jednoho grafu

Linespec - specifikace křivky

plot(X1,Y1,LineSpec1,X2,Y2,LineSpec2,...)
více veličin do jednoho grafu s různými nastaveními

Příklad pro vykreslení grafu:

```
x1=[2 4 3; 6 4 2];  
x2=[3 5 4; 2 7 1];  
y1=[2 5 8; 7 4 9];  
y2=[6 1 7; 1 2 5];  
plot(x1,y1,x2,y2)
```

f) Vykreslení čar a bodů v 3D prostoru

plot3 - syntaxe pro vykreslení čar a bodů v 3D prostoru

plot3(X1,Y1,Z1,X2,Y2,Z2,...)
více veličin do jednoho grafu

Linespec - specifikace křivky

plot3(X1,Y1,Z1,LineSpec1,X2,Y2,Z2,LineSpec2,...)
více veličin do jednoho grafu s různými nastaveními

Příklad pro vykreslení grafu:

```
x1=[2 4 3; 6 4 2];  
x2=[3 5 4; 2 7 1];  
y1=[2 5 8; 7 4 9];  
y2=[6 1 7; 1 2 5];  
z1=[1 4 7; 5 2 9];  
z2=[7 6 2; 2 7 4];  
plot3(x1,y1,z1,x2,y2,z2)  
grid on
```

g) Vykreslení grafu s osou y vlevo i vpravo

plotyy - syntaxe pro vykreslení grafu s osou y vlevo i vpravo

plotyy(X1,Y1,X2,Y2)
více veličin do jednoho grafu

Linespec - specifikace křivky

plotyy(X1,Y1,LineSpec1,X2,Y2,LineSpec2)
více veličin do jednoho grafu s různými nastaveními

Příklad pro vykreslení grafu:

```
x=0:0.01:20;
y1=200*exp(-0.05*x).*sin(x);
y2=0.8*exp(-0.5*x).*sin(10*x);
[ax,h1,h2]=plotyy(x,y1,x,y2,plot)
set(get(ax(1),ylabel),string,Levá y-ová osa)
set(get(ax(2),ylabel),string,Pravá y-ová osa)
xlabel(Nula až 20\musec.)
title(Oznacení plotyy)
```

Zde se setkáváme i s pojmenováním jednotlivých os

h) Umístění více grafů vedle sebe do jednoho okna

subplot - syntaxe pro umístění více grafů vedle sebe do jednoho okna
subplot(řádek,sloupec,číslo_grafu)

subplot(řádek,sloupec,číslo_grafu,replace) - nahradí vybraný graf

Příklad pro vykreslení grafu:

```
income=[3.2 4.1 5.0 5.6];
outgo=[2.5 4.0 3.35 4.9];
subplot(2,1,1);plot(income)
subplot(2,1,2);plot(outgo)
```

1.4.2 Popisky grafů

Cílem popisků grafů je seznámit uživatele s:

- a) Označením vrstevnic vrstevnicového grafu
- b) Datově formátovanými značkami u os
- c) Zobrazení legendy grafu pro jednotlivé křivky
- d) Nadpisem a popisky jednotlivých os

a) Označení vrstevnic vrstevnicového grafu

clabel(C,v) - označení vrstevnic vrstevnicového grafu
v- vektor

příklad pro vykreslení grafu:

```
[x,y]=meshgrid(-2:.2:2);
z=x.^exp(-x.^2-y.^2);
[C,h]=contour(x,y,z);
clabel(C,h);
```

b) Datově formátované značky u os

datetick(osa) - datově formátované značky u os
datetick(osa,tvar_značek)

Příklady:

<i>Tvar značek(číslo)</i>	<i>tvar značek(řetězec)</i>	<i>příklad</i>
0	dd-mmm-rrrr HH:MM:SS	01-Mar-2000 15:45:17
1	dd/mmm/rrrr	01-Mar-2000
2	mm/dd/rr	03/01/00
3	mmm	Mar
4	m	M
5	mm	03
6	mm/dd	03/01
7	dd	01
8	ddd	Wed
9	d	W
10	rrrr	2000
11	rr	00
.	.	.
.	.	.
.	.	.

a dále by bylo možné ještě pokračovat, ale na ukázkou to stačí

Příklad pro vykreslení grafu:

Graf přibližné populace USA v časovém období 1900-1990

```
t=(1900:10:1990);
```

```
p=[75000 91000 105000 123000 131000 150000 179000 203000 226000  
249000];
```

```
plot(datetime(t,1,1),p) - převod roků na datový formát a vykreslení  
grid on
```

```
datetick('x',11) - nahrazení x-ové popisky posledního dvojčíslí roku
```

c) Zobrazení legendy grafu pro jednotlivé křivky

legend - zobrazení legendy grafu pro jednotlivé křivky

```
legend('křivka1','křivka2',...)
```

```
legend(matice řetězců popisů)
```

```
legend('off') - legenda vypnuta
```

```
legend('hide') - legenda tam je, ale je neviditelná
```

```
legend('show') - legenda je ukázaná
```

```
legend('box_on')
```

```
legend('box_off')
```

```
legend(...,pozice)
```

```
pozice      význam
```

```
-1          vedle vykreslené oblasti vpravo
```

```
0          umístění legendy tak, aby zakrývala co nejméně bodů
```

```
1          legenda v horním pravém rohu os
```

```
2          legenda v levém horním rohu
```

```
3          legenda v dolním levém rohu
```

```
4          legenda v dolním pravém rohu
```

Příklad pro vykreslení grafu:
`x=-pi:pi/20:pi;`
`plot(x,cos(x),'-ro',x,sin(x),'-b')`
`h=legend('cos','sin',2);`

d) Nadpis a popisky jednotlivých os

title(osa) - nadpis grafu

`title('název grafu')`
`title(funkce)`

xlabel - popis osy x

ylabel - popis osy y

zlabel - popis osy z

Příklad pro vykreslení grafu:
`x=0:0.01:20;`
`y1=200*exp(-0.05*x).*sin(x);`
`y2=0.8*exp(-0.5*x).*sin(10*x);`
`[ax,h1,h2]=plotyy(x,y1,x,y2,plot)`
`set(get(ax(1),ylabel),string,Levá y-ová osa)`
`set(get(ax(2),ylabel),string,Pravá y-ová osa)`
`xlabel(Nula až 20\musec.)`
`title(Oznacení plotyy)`

Zde se setkáváme i s pojmenováním jednotlivých os

1.4.3 Specializované grafy

Pro jejich velké zastoupení byly specializované grafy rozděleny do tří skupin.

1.4.3.1 Část I.

Cílem 1.části je seznámit uživatele se specializovanými grafy a to:

- a) Plošný graf
- b) Sloupcový graf
- c) 3D sloupcový graf
- d) Koláčový graf
- e) 3D koláčový graf
- f) Vrstevnicový graf
- g) 3D vrstevnicový graf
- h) Vyplněný vrstevnicový graf
- i) Kometový graf
- j) 3D kometový graf
- k) Kompasový graf
- l) Graf vektorů rychlosti

a) Plošný graf

area(X,Y) - příkaz pro vykreslení grafu

area(Y) - vektor nebo suma sloupců matice

area(...,ymin) - nejnižší hodnota y v grafu

Příklad pro vykreslení grafu:

```
y=[1,5,3;3,2,7;1,5,3;2,6,1];
```

```
area(y)
```

```
grid on
```

```
colormap summer
```

```
set(gca,'Layer','top') - abz se nepřekrývaly průběhy
```

```
title 'Stacked Area Plot'
```

b) Sloupcový graf

bar(Y) - příkaz pro vykreslení grafu

bar(X,Y) - X je rostoucí vektor, pozice bodů Y

bar(...,šířka) - nastavit šířku obdelníků

bar(...,styl) - group-pro více průběhů jsou obdelníky vedle sebe
stack-obdelníky přes sebe

bar(...,LineStyle) - nastavit barvu obdelníku

barh() - vodorovný sloupcový graf

Příklad pro vykreslení grafu:

```
y=round(rand(5,3)*10);
```

```
subplot(2,2,1)
```

```
bar(y,'group')
```

```
title 'group'
```

```
subplot(2,2,2)
```

```
bar(y,'stack')
```

```
title 'stack'
```

```
subplot(2,2,3)
```

```
barh(y,'stack')
```

```
title 'stack'
```

```
subplot(2,2,4)
```

```
bar(y,1.5)
```

```
title 'width=1.5'
```

c) 3D sloupcový graf

bar3(Y) - příkaz pro vykreslení grafu

bar3(X,Y) - X je rostoucí vektor, pozice bodů Y

bar3(...,šířka) - nastavit šířku obdelníku

bar3(...,styl) - detached
grouped

stacked
bar(...,LineSpec) - nastavit barvu obdelníků
barh3() - vodorovný sloupcový graf
Příklad pro vykreslení grafu lze použít z předešlé úlohy s doplněním na 3D

d) Koláčový graf

pie(X) - příkaz pro vykreslení grafu
pie(X,explode) - rozdělí jednotlivé části

Příklad pro vykreslení grafu:
x=[3 1 0.5 2.5 2]; - procenta v koláči
explode=[0 1 0 0 0];
pie(x,explode)
colormap jet

e) 3D koláčový graf

pie3(X) - příkaz pro vykreslení grafu
pie3(X,explode) - rozdělí jednotlivé části

Příklad pro vykreslení grafu lze použít z předešlé úlohy s doplněním na 3D

f) Vrstevnicový graf

contour(z) - matici z jako průmět do roviny xy
contour(z,n) - n počet urovní vrstevnic
contour(x,y,z) - x,y vymezují hranice os
contour(...,LineSpec) - nastaví barvy

Příklad pro vykreslení grafu:
Nakreslete vrstevnicový graf fce:
 $z = xe^{(-x^2 - y^2)}$
v intervalu $-2 \leq x \leq 2$ $-2 \leq y \leq 2$

```
[x,y]=meshgrid(-2:2:2,-2:2:3);  
z=x.*exp(-x.^2-y.^2);  
[c,h]=contour(x,y,z);  
clabel(c,h)  
colormap cool
```

g) 3D vrstevnicový graf

contour3(z) - matici z jako průmět do roviny xy
contour3(z,n) - n počet urovní vrstevnic
contour3(x,y,z) - x,y vymezují hranice os
contour3(...,LineSpec) - nastaví barvy

Příklad pro vykreslení grafu lze použít z předešlé úlohy s doplněním na 3D

h) Vyplněný vrstevnicový graf**ezcontour** - vrstevnice pomocí funkce dvou proměnných

ezcontour(f) - f je řetězec matematické fce obsahující 2 proměnné

ezcontour(f,domena) domena - [min,max]

ezcontour(...,n) n - hustota sítě v bodech

ezcontourf - vyplněný vrstevnicový graf fce dvou proměnných

Příklad pro vykreslení grafu:

f = [3*(1-x)^2*exp(-(x^2)-(y+1)^2),...

` - 10\*(x/5 - x^3 - y^5)\*exp(-x^2-y^2)` ,...

` - 1/3\*exp(-(x+1)^2 - y^2)`];

ezcontourf(f,[-3,3],49)

i) Kometový graf – postupné vykreslování ve 2 D**comet(y)** - vektor y

comet(x,y) - vektor y na vektoru x

Příklad pro vykreslení grafu:

t = 0:.01:2*pi;

x = cos(2*t).*(cos(t).^2);

y = sin(2*t).*(sin(t).^2);

comet(x,y);

j) Kometový graf – postupné vykreslování ve 3D**comet3(z)** - vektor z

comet3x,y,z) - v souřadnicích x,y,z

Příklad pro vykreslení grafu:

t = -10*pi:pi/250:10*pi;

comet3((cos(2*t).^2).*sin(t),(sin(2*t).^2).*cos(t),t);

k) Kompasový graf**compass(x,y)** - příkaz pro vykreslení grafu

compass(z) x,y,z jsou kartézské souřadnice

compass(...,LineStyle) - pro nastavení parametrů

Příklad pro vykreslení grafu:

Z = eig(randn(20,20));

compass(Z)

l) Graf vektorů rychlosti**fetaher(U,V)** - x-ové složky vektorů rychlostí - U

y-ové složky vektorů rychlostí - V
 feather(z) - z - vektor komplexních čísel reprezentujících rychlosti
 feather(...,LineSpec) - parametry čáry grafu

Příklad pro vykreslení grafu:
 theta = (-90:10:90)*pi/180;
 r = 2*ones(size(theta));
 [u,v] = pol2cart(theta,r);
 feather(u,v);

1.4.3.2 Část II.

Cílem 2.části je seznámit uživatele se specializovanými grafy a to:

- a) Grafem rychlosti
- b) 3D grafem rychlosti
- c) Stopkovým grafem
- d) 3D stopkovým grafem
- e) Schodišťovým grafem
- f) Síťovým grafem
- g) Síťovým konturovaným grafem
- h) Snadným vykreslením funkce
- i) Snadným vykreslením 3D křivky
- j) Grafem v polárních souřadnicích
- k) Grafem pro vykreslení barevné plochy
- l) Kombinací plochy, povrchu s konturami

a) Graf rychlosti

quiver(X,Y,U,V) - vektor rychlosti reprezentovaný složkami U a V v bodě X,Y
 quiver(...,LineSpec) - parametry čáry grafu

Příklad pro vykreslení grafu:
 [X,Y] = meshgrid(-2:.2:2);
 Z = X.*exp(-X.^2 - Y.^2);
 [DX,DY] = gradient(Z,.2,.2);
 contour(X,Y,Z)
 hold on
 quiver(X,Y,DX,DY)
 colormap hsv
 grid off
 hold off

b) 3D graf rychlosti

quiver3(X,Y,Z,U,V,W) - vektor rychlosti reprezentovaný složkami U,V a W v bodě X,Y,Z
 quiver3(X,Y,Z,U,V,W,měřítko) - měřítko je délka šipky vektoru
 quiver(...,LineSpec) - parametry čáry grafu

Příklad pro vykreslení grafu:

```
[X,Y] = meshgrid(-2:0.25:2,-1:0.2:1);
Z = X.* exp(-X.^2 - Y.^2);
[U,V,W] = surfnorm(X,Y,Z);
quiver3(X,Y,Z,U,V,W,0.5);
hold on
surf(X,Y,Z);
colormap hsv
view(-35,45)
axis ([-2 2 -1 1 -.6 .6])
hold off
```

c) Stopkový graf

stem(y) - vektor y

stem(x,y) - y v závislosti na x

stem(...,fill) - vyplnění koleček na konci stopky

stem(...,LineStyle) - parametry čáry grafu

Příklad pro vykreslení grafu:

```
y = linspace(0,2*pi,10);
h = stem(cos(y),'fill','-');
set(h(3),'Color','r','LineWidth'
axis ([0 11 -1 1])
```

d) 3D stopkový graf

stem3(z) - vektor z

stem3(x,y,z) - z v závislosti na x a y

stem3(...,fill) - vyplnění koleček na konci stopky

stem3(...,LineStyle) - parametry čáry grafu

Příklad pro vykreslení grafu:

```
X = linspace(0,1,10);
Y = X./2;
Z = sin(X) + cos(Y);
stem3(X,Y,Z,'fill')
view(-25,30)
```

e) Schodišťový graf

stairs(y) - vektor y

stairs(x,y) - y v závislosti na x

[xb,yb]=stairs(x,y) - vrací vektor schodů,nic nekreslí

stairs(...,LineStyle) - parametry čáry grafu'

Příklad pro vykreslení grafu:

```
x = 0:.25:10;
stairs(x,sin(x))
```

f) Síťový graf

ezmesh(f) - f je řetězec fce dvou proměnných

ezmesh(f,domena)

ezmesh(...,n) - síť $n \times n$

ezmesh(...,circ) - vykreslí f přes disk vystředěný uvnitř domény

Příklad pro vykreslení grafu:

ezmesh('x*exp(x^2+y^2)',40)

colormap([0 0 1])

g) Síťový konturovaný graf

ezmeshc(f) - f je řetězec fce dvou proměnných

ezmeshc(f,domena)

ezmeshc(...,n) - síť $n \times n$

ezmeshc(...,circ) - vykreslí f přes disk vystředěný uvnitř domény

Příklad pro vykreslení grafu:

ezmeshc('y/(1 + x^2 + y^2)',[-5,5,-2*pi,2*pi])

h) Snadné vykreslení funkce

ezplot(f) - f je řetězec fce dvou proměnných

ezplot(f,[min,max])

ezplot(f,[xmin,xmax,ymin,ymax])

ezplot(...,figure) - vykreslí do zadaného okna figure

Příklad pro vykreslení grafu:

ezplot('x^2-y^4')

i) Snadné vykreslení 3D křivky

ezplot3(x,y,z)

ezplot3(x,y,z,[tmin,tmax])

ezplot3(...,'animate') - animovaná stopa křivky

Příklad pro vykreslení grafu:

ezplot3('sin(t)','cos(t)','t',[0,6*pi],'animate')

j) Graf v polárních souřadnicích

ezpolar(f) - vykreslí graf fce

ezpolar(f[a,b]) - $a < \theta < b$

Příklad pro vykreslení grafu:

ezpolar('1+cos(t)')

k) Graf pro vykreslení barevné plochy

ezsurf(f) - vykreslí graf fce

`ezsurf(x,y,z)`

`ezsurf(...,n)`

`ezsurf(...,circ)` - význam podobný jako u příkazu `ezmesh`

!!!!Nevykresluje body,kde není fce definována
je třeba takovou fci upravit pomocí příkazu `real`

Příklad pro vykreslení grafu:

`ezsurf('real(atan(x+i*y))')`

l) Kombinací plochy, povrchu s konturami

ezsurf(f) – vykreslí graf fce

`ezsurf(x,y,z)'`

`ezsurf(...,n)'`

`ezsurf(...,circ)` - význam podobný jako u příkazu `ezmesh`

Příklad pro vykreslení grafu:

kombinace graf povrchu a vrstevnice

`ezsurf('y/(1 + x^2 + y^2)',[-5,5,-2*pi,2*pi],35)`

1.4.3.3 Část III.

Cílem 3.části je seznámit uživatele se specializovanými grafy a to:

- a) Histogramem
- b) Úhlovým histogramem
- c) Válcem
- d) Grafem pseudobarevných ploch
- e) Žebrovanými plochami
- f) Volumetrickým plátovým grafem
- g) Koulí
- h) Vodopádovou plochou
- i) 2D graf rozložení prvků matice
- j) 3D graf rozložení prvků v poli
- k) Vyplněné 2D polygony
- l) Vyplněné 3D polygony

a) Histogram

n=hist(Y) - dá vektor Y do 10 zásobníků rovnoměrně oddělených

a vrátí počet prvků v každém zásobníku

`n=hist(Y,x)` - vrátí distribuci Y podél délky x

`n=hist(Y,nbins)` - nbins- počet prvků zásobníku

Příklad pro vykreslení grafu:

histogram zvonovité křivky Gaussova rozdělení

`x = -2.9:0.1:2.9;`

`y = randn(10000,1);`

`hist(y,x)`

b) Úhlový histogram

rose(theta) - úhel po 20 stupních

rose(thrta,x) - vektor x specifikuje počet úhlů a jejich velikost

rose(theta,nbins) - počet rovnoměrných úhlů

n=hist(Y,nbins) - nbins- počet prvků zásobníku

Příklad pro vykreslení grafu:

```
theta = 2*pi*rand(1,50);
```

```
rose(theta)
```

c) Válec

cylinder(r) - r je obrysová křivka

Příklad pro vykreslení grafu:

```
cylinder
```

```
axis square
```

```
h = findobj('Type','surface');
```

```
set(h,'CData',rand(size(get(h,'CData'))))
```

d) Graf pseudobarevných ploch

pcolor(colormap) - r je obrysová křivka

pcolor(X,Y,colormap) - v bodech specifikovaných X a Y

Příklad pro vykreslení grafu:

```
pcolor(hadamard(20))
```

```
colormap(gray(2))
```

```
axis ij
```

```
axis square
```

e) Žebrované plochy

ribbon(Y)

```
ribbon(X,Y)
```

ribbon(X,Y,šířka) - šířka=šířka žeber

Příklad pro vykreslení grafu:

```
[x,y] = meshgrid(-3:.5:3,-3:.1:3);
```

```
z = peaks(x,y);
```

```
ribbon(y,z)
```

```
colormap hsv
```

f) Volumetrický plátový graf

slice(v,sx,sy,sz) - pláty ve směru x,y,z, velikost v bodech sx,sy,sz

slice(x,y,z,r,sx,sy,sz) - kde x,y,z představují souřadnice

slice(...,'metoda') - určuje metodu interpolace(linear,cubic,nearest)

Příklad pro vykreslení grafu:

```
[x,y,z] = meshgrid(-2:.2:2,-2:.25:2,-2:.16:2);
v = x.*exp(-x.^2-y.^2-z.^2);
xslice = [-1.2,.8,2]; yslice = 2; zslice = [-2,0];
slice(x,y,z,v,xslice,yslice,zslice)
colormap hsv
```

g) Koule

sphere(N) - koule o N rovnoběžkách a N polednicích

Příklad pro vykreslení grafu:

```
sphere
axis equal
```

h) Vodopádovou plochou

waterfall(z)

```
waterfall(x,y,z)
waterfall(...,Colormap)
```

Příklad pro vykreslení grafu:

```
[X,Y,Z] = peaks(30);
waterfall(X,Y,Z)
```

i) 2D graf rozložení prvků matice

scatter(x,y,s,c) - v souřadnicích zadávaných vektory x,y kreslí kolečka zadaná vektorem s,v % barvě c zadané pro každé kolečko jinak

```
scatter(x,y)'
scatter(...,typ kolečka) - podle specifikace LineSpec
scatter(...,filled) - vyplněná kolečka
```

Příklad pro vykreslení grafu:

```
load seamount
scatter(x,y,5,z)
```

j) 3D graf rozložení prvků v poli

scatter3(x,y,z,s,c) - v souřadnicích zadávaných vektory x,y,z kreslí kolečka zadaná vektorem s,v % barvě c zadané pro každé kolečko jinak

```
scatter3(x,y,z)
scatter3(...,typ kolečka) - podle specifikace LineSpec
scatter3(...,filled) - vyplněná kolečka
```

Příklad pro vykreslení grafu:

```
[x,y,z] = sphere(16);
X = [x(:)*.5 x(:)*.75 x(:)];
Y = [y(:)*.5 y(:)*.75 y(:)];
Z = [z(:)*.5 z(:)*.75 z(:)];
```

```
S = repmat([1 .75 .5]*10,prod(size(x)),1);
C = repmat([1 2 3],prod(size(x)),1);
scatter3(X(:),Y(:),Z(:),S(:),C(:),'filled'), view(-60,60)
```

k) Vyplněné 2D polygony

fill(X,Y,C) - vytvoří vyplněný polygon z X,Y, v v C je 3 složkový vektor
fill(...,ColorSpec)

Příklad pro vykreslení grafu:
vytvoření červeného osmiúhelníku
t = (1/16:1/8:1)*2*pi;
x = sin(t);
y = cos(t);
fill(x,y,'r')

l) Vyplněné 3D polygony

fill3(X,Y,Z,C) - vytvoří vyplněný polygon z X,Y,Z, v v C je 3 složkový vektor
fill3(...,ColorSpec)

Příklad pro vykreslení grafu:
vytvoření červeného osmiúhelníku
X = [0 1 1 2;1 1 2 2;0 0 1 1];
Y = [1 1 1 1;1 0 1 0;0 0 0 0];
Z = [1 1 1 1;1 0 1 0;0 0 0 0];
C = [0.5000 1.0000 1.0000 0.5000;
1.0000 0.5000 0.5000 0.1667;
0.3330 0.3330 0.5000 0.5000];
fill3(X,Y,Z,C)

1.5 Jednoduché úlohy pro MATLAB – Simulink

Simulink je grafickou nástavbou programu MATLAB. Jedná se o řešení soustavy nelineárních diferenciálních rovnic s grafickým zadáváním řešené soustavy, připomínající zapojení na analogovém počítači tzv „modely“. Simulink umožňuje graficky sledovat průběhy veličin v libovolném místě zapojení. Simulink spustíme pomocí tlačítka „Spustit Simulink“.

Dále je třeba říci, že samotný program, který je vytvořen pro tuto bakalářskou práci, obsahuje v sekci „Jednoduché úlohy pro MATLAB Simulink“ elementární úlohy týkající se teorie řízení:

1. Uzavřený regulační obvod s PID regulátorem
2. Uzavřený regulační obvod s PI regulátorem

3. Regulační obvod s využitím Smithova prediktoru

Všechny tři zmiňované úlohy jsou popsány v části 1.5 *Simulink*, v kapitole 1.5.3.

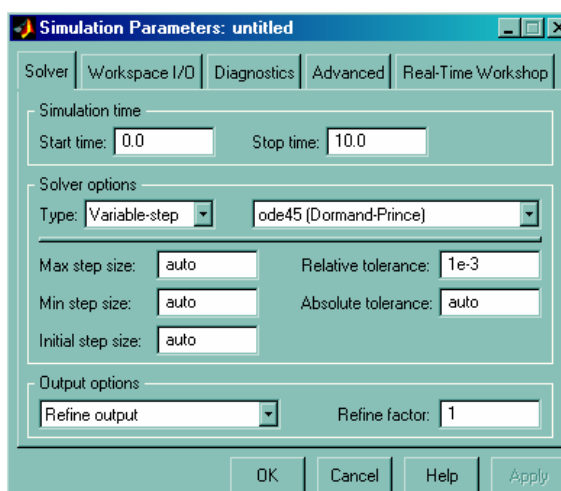
1.5.1 Vytvoření modelu a jeho spuštění v prostředí Simulink

Vlastní model se vytváří výběrem bloků z knihoven a jejich přesunem do okna modelu. Kromě přesunu z knihovny je možné i bloky v okně modelu standardním způsobem kopírovat. Po přesunu lze zadat parametry jednotlivých bloků. Název nového bloku se nastaví automaticky tak, aby byl v rámci okna modelu jednoznačný. Např. první přesunutý blok „SUM“ bude mít název „SUM“, druhý poté „SUM 1“. Tyto názvy je možné měnit pomocí editace názvu bloku.

Dále je třeba spojit odpovídající vstupy a výstupy bloků. Výstup jednoho bloku může být připojen na libovolný počet vstupů jiných bloků nebo toho samého bloku. V libovolném místě okna lze možno umístit komentář, vysvětlující text.

Dvojklikem na požadovaném místě lze přejít do režimu editace.

Po vytvoření modelu nastavíme základní parametry simulace. Po otevření nabídky *Simulation* z hlavního panelu okna modelu a výběru položky *Parameters* se objeví okno parametrů simulace se záložkami *Solver*, *Workspace I/O*, *Diagnostic*, *Advanced*, *Real – Time Workshop*.



Obr. 2 Parametry simulace MATLAB Simulink

Pod záložkou *Solver* je možné nastavit čas simulace, volbu řešitele a volby týkající se požadavků na okamžiky, kdy má být k dispozici hodnota výstupu.

Volby záložky *Workspace I/O* se týkají možností napojení na pracovní prostor MATLABu. Je možné předepsat, že mají přebírat počáteční stavy, požadované okamžiky výpočtu a průběhy vstupních veličin z proměnných v pracovním prostoru programu MATLAB.

Volba záložky *Diagnostic* umožňuje nastavení, zda události mají vyvolávat hlášení a na jaké úrovni.

Nyní je možné spustit simulaci. V případě, že nebyl zařazen žádný blok pro vizualizaci průběhu výsledku se pouze ve stavovém řádku okna modelu ukazuje průběh času výpočtu. V případě, že použijeme některý zobrazovací blok, je potřeba dvojklikem na tento blok otevřít zobrazovací okno.

1.5.2 Standardní knihovny prostředí Simulink

Každá knihovna obsahuje jednotlivé bloky. Většina bloků dovoluje zadat hodnoty parametrů. Bloky jsou v knihovnách seskupeny podle oblasti použití.

Ukažme si nyní několik nejčastěji používaných knihoven v prostředí Simulink.

V knihovně „**source**“ jsou bloky použitelné jako zdroje běžných signálů.

V knihovně „**sinks**“ jsou bloky pro zobrazení a uložení průběhů signálů a blok umožňující ukončit simulaci z modelu.

Knihovna „**linear**“ obsahuje bloky sloužící pro vytváření lineárních modelů. Tyto prvky tvoří obvykle základ dynamického modelu.

Knihovna „**discrete**“ obsahuje bloky potřebné pro tvorbu diskretních popisů soustav. Kromě bloků víceméně ekvivalentních blokům z knihovny „**linear**“ jsou zde i takové, které obsahují tvarovače nultého řádu.

Poslední standardní knihovnou je „**connections**“. Tato knihovna obsahuje bloky pro slučování a rozdělování signálů („**mux**“ a „**demux**“), ukládání a čtení do proměnných programu MATLAB a další pomocné bloky.

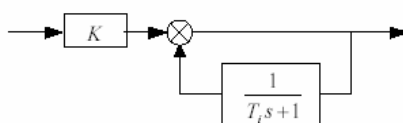
1.5.3 PI regulátor, PID regulátor, Smithův prediktor

PI regulátor

Zákon řízení proporcionálně integračního (PI) regulátoru v lineární oblasti je dán vztahem :

$$u(t) = K \frac{1}{T} \int_0^t e(\tau) d\tau \quad (1)$$

kde „ K “ je zesílení a „ T “ integrační časová konstanta regulátoru. Díky integrátoru má tento regulátor velmi žádanou vlastnost, a to – nulovou regulační odchylku v ustáleném stavu při konstantní požadované hodnotě „ w “ a poruchách „ d, n “. Kdyby tomu tak nebylo, docházelo by v důsledku integrace nenulové odchylky ke změně výstupu regulátoru „ u “ a uzavřená smyčka by nebyla v ustáleném stavu. Na obrázku (Obr.3) je zobrazena jedna z prvních implementací PI regulátoru.



Obr. 3 Princip PI regulátoru

PID regulátor

Snaha vylepšit stabilitu uzavřené smyčky s PI regulátorem vedla k zavedení derivační (D) složky. Ideální PID regulátor realizuje tedy zákon řízení ve tvaru:

$$u(t) = K(e(t) \frac{1}{T} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt}) \quad (2)$$

kde parametr T_d se nazývá derivační časová konstanta. Složku PD můžeme chápat jako modifikovanou P složku, kde je odchylka $e(t)$ nahrazena predikovanou odchylkou $e(t+T_d)$, neboť z Taylorova rozvoje plyne:

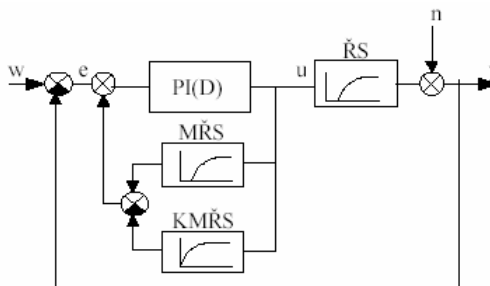
$$e(t+T_d) = e(t) + T_d \frac{de(t)}{dt} \quad (3)$$

Derivační složka umožňuje tedy předvídat budoucí chování procesu. Někdy je však nutné omezení derivační složky, a to v případech, kdy řízený systém obsahuje velké dopravní zpoždění nebo kdy je regulovaná veličina zatížena velkým šumem měření.

Smithův prediktor

Pro špatně regulované systémy ($\tau > 0,4$) nemusí být kvalita řízení dosažena PI(D) regulátorem vyhovující. V případě, že řízený systém obsahuje dominantní dopravní zpoždění, nebo je neminimálně fázový, lze kvalitu řízení podstatným

způsobem vylepšit doplněním PI(D) regulátoru kladnou zpětnou vazbou, která obsahuje model řízeného systému. Toto vylepšení je naznačeno na obrázku (Obr. 4) Zkratka MŘS označuje model řízeného systému obsahující dopravní zpoždění. Kompenzovaný model řízeného systému (KMŘS) získáme z MŘS jednoduše vypuštěním jeho dopravního zpoždění.



Obr. 4 Princip Smithova prediktoru, struktura s PI(D) regulátorem

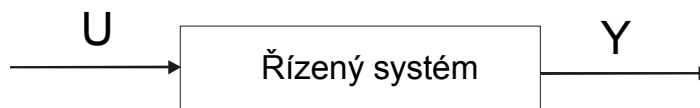
1.6 Využití CONTROL TOOLBOXu

Pod pojmem řízený systém budeme rozumět orientovaný objekt s jedním vstupem a jedním výstupem (Obr.5). Z důvodu jednoduchosti budeme dále předpokládat, že systém nemění své vlastnosti v čase (t - invariantnost), a že okamžitá hodnota jeho výstupu $y(t_0)$ v okamžiku t_0 závisí pouze na hodnotách vstupní funkce $u(t)$, $t \leq t_0$.

Na systém se tedy můžeme dívat jako na jistý operátor, který libovolné vstupní funkci u přiřazuje jistou výstupní funkci y .

Tuto skutečnost budeme zapisovat následovně:

$$y = L \cdot u \quad (4)$$



Obr. 5 Objekt s jedním vstupem a jedním výstupem

1.6.1 Přechodová charakteristika

Jestliže na vstup systému přivedeme vstupní funkci ve tvaru jednotkového skoku, potom na výstupu obdržíme přechodovou charakteristiku $h(t)$ danou vztahem:

$$h(t) = \frac{Y(s)}{U(s)} \quad (5)$$

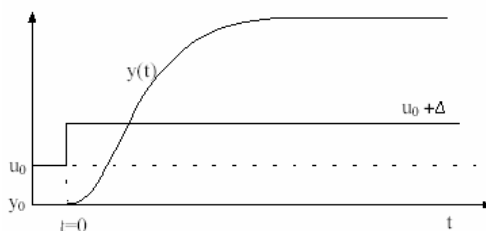
$$y(t) = G(s) \cdot U(s) = G(s) \cdot \frac{1}{s} \quad (6)$$

V případě reálného procesu můžeme přechodovou charakteristiku experimentálně získat následovně:

1. Vyčkat na ustálený stav procesu ($u(0)=u_0$; $y(0)=y_0$).
2. Změřit skokem vstup u z konstantní hodnoty u_0 na hodnotu $u_0+\Delta$
3. Odměřit odezvu systému $y(t)$
4. Přechodovou charakteristiku nyní určit ze vztahu :

$$g(t) = \frac{1}{\Delta} (y(t) - y_0) \quad (7)$$

Je vhodné volit skok Δ co možná největší, abychom obdrželi dobrý poměr užitečného signálu k šumu a poruchám.

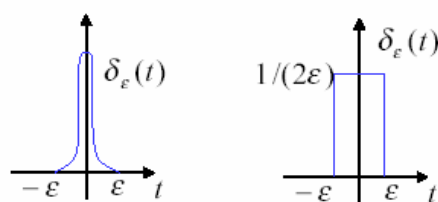


Obr. 6 Měření přechodové charakteristiky

1.6.2 Impulsní charakteristika

Jestliže na vstup systému přivedeme Diracův impuls, na výstupu bude Impulsní charakteristika. Impulsní charakteristika je tedy odezva systému na Diracův impuls.

Diracův impuls je demonstrován na následujícím obrázku (Obr. 7):

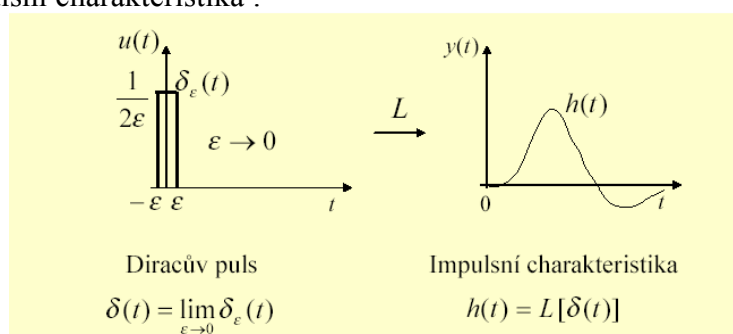


$$\delta(t) = \lim_{\varepsilon \rightarrow 0} \delta_{\varepsilon}(t)$$

$$\delta(t) = \begin{cases} \infty & t = 0 \\ 0 & t \neq 0 \end{cases}$$

Obr. 7 Diracův impuls

Impulsní charakteristika :



Obr. 8 Impulsní charakteristika

Většinou nastane problém při měření impulsní charakteristiky, a to takový, že charakteristika se nedá přímo přesně odměřit, neboť Diracův impuls má nekonečnou amplitudu.

1.6.3 Amplitudově fázová frekvenční charakteristika

Z principu superpozice vyplývá další vlastnost lineárních systémů. Je to odezva systému na sinusový vstup.

$u(t) = \sin \omega t$ je v ustáleném stavu opět sinusová funkce.

Reálná proměnná ω je frekvenční přenos systému. Způsob experimentálního určení jednoho jediného bodu je :

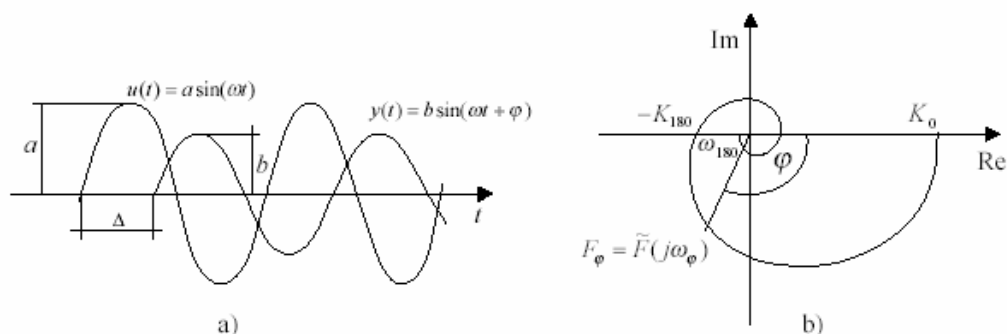
1. Připojit na vstup sinusový signál s danou frekvencí a vhodně vybranou amplitudou a .
2. Vyčkat odeznění přechodového děje a poté odměřit amplitudu b a časové posunutí Δ výstupu.
3. Určit bod $F(j\omega)$ komplexní frekvenční charakteristiky příslušný frekvenci ω podle vztahu:

$$F(j\omega) = \frac{b}{a} e^{-j\omega} \quad (8)$$

kde

$$\varphi = \frac{\Delta}{T} 2\pi \quad (9)$$

Je zřejmé, že odměření celé frekvenční charakteristiky je velmi zdoluhavé. Pro návrh regulátoru stačí vybrat jeden nebo dva body této charakteristiky.



Obr. 9 Frekvenční charakteristika. a) záznam vstupu a výstupu systému. b) charakteristika v komplexní rovině

1.7 Audio, video a práce s obrázky

Závěrečná kapitola je zaměřena na práci s audiem, videem a na práci s obrázky v MATLABU. Jednotlivé funkce nám objasní činnosti audiopřehrávače, záznamů zvuku, načtení a uložení audio snímků do souboru, práci s AVI souborem, možnosti jednotlivých obrázků atd.

1.7.1 Obecné audio funkce

audioplayer - vytvoří objekt audiopřehrávače

audiorecorder - vytvoří objekt záznamu zvuku

beep - pípnutí

beep - pípne, pokud je povoleno

beep on - povolení pípání

beep off - zakázání pípání

s=beep - řekne, zda je povoleno pípání nebo zakázáno

lin2mu - převod lineárního audisignálu na mu standard

mu=lin2mu(y)

mu2lin - převod mu-standartního signálu na lineární
`y=mu2lin(mu)`

sound - převod vektoru na zvuk
`sound(y,frekvence,bity)` - pouze pro OS Windows

soundsc - zařazení dat do stupnice a přehrání
`soudnsc(y,frekvence,bity)` - pouze pro OS Windows

Příklady

- 1) `load handel;`
`player=audioplayer(y,Fs);`
`play(player,[1 (get(player,'SampleRate')*3)]);`
`stop(player);`
- 2) `recorder = audiorecorder(44100,16,2);`
`recordblocking(recorder,3.5);`
`audioarray = getaudiodata(recorder);`
 - pomocí mikrofону zaznamená 3,5s data o frekvenci 44,1 kHz, 16 bit, stereo a pak vrátí tyto data jako pole double
- 3) pomocí mikrofону zaznamená 8 bit, 22 kHz mono dat, přehraje je a vrátí do prostředí Matlabu jako pole mint 8
`micrecorder=audiorecorder(22050,8,1);`
`record(micrerecorder);` - nyní mluvte na mikrofón
`stop(micrerecorder);`
`speechplayer=play(micrerecorder);` - poslech vaší řeči
`stop(speechplayer)`
`speechdata=getaudiodata(micrecorder,'mint8')`

1.7.2 Funkce pro Microsoft WAVE

wavpaly - přehrání záznamového zvuku na PC
`wavpaly(y,frekvence)`

wavread - načtení *.wav souboru
`y=wavread('soubor')`

wavrecord - záznam zvuku pomocí PC audio vstupu
`y=wavrecord(počet_vzorků,frekvence)`
`počet_vzorků` - zaznamená počet vzorků při dané frekvenci

wavwrite - zápis *.wav souboru
`wavwrite(y,'soubor')`
`wavwrite(y,frekvence,'soubor')`
`wavwrite(y,frekvence,bity,'soubor')`
`bity` - max 32

Příklady

- 1) load chirp;
y1 = y; Fs1 = Fs;
load gong;
wavplay(y1,Fs1,'sync')
wavplay(y,Fs)
- 2) Fs = 11025;
y = wavrecord(5*Fs,Fs,'int16');
wavplay(y,Fs);
- zaznamená 5 sekund 16 bitového audio signálu o frekvenci 11,025 kHz
a přehraje tento zvuk

1.7.3 AVI funkce

addframe - přidání okna do avi souboru

avionjekt=addfreme(aviobjekt,ram1,ram2,...)
přidá příslušný počet rámců do avi souboru

avifile - vytvoření nového AVI souboru

aviinfo - vrácení informací o AVI souboru
informace=aviinfo(soubor)

aviread - načtení AVI souboru

close - zavření AVI souboru

movie2avi - vytvoření *.avi souboru z videa Matlabu
souborů identifikovaného pomocí avi objekt
movie2avi(videosoubor)

1.7.4 Práce s obrázky

im2java - převod obrázku na Java obrázek
jimage=im2java(R(mB))

iminfo - informace o grafickém souboru

imread - načtení obrázku z grafického souboru

imwrite - zápis obrázku do grafického souboru

close - zavření AVI souboru

movie2avi - vytvoření *.avi souboru z videa Matlabu
souborů identifikovaného pomocí avi objekt
movie2avi(videosoubor)

Do kategorie obrázků patří všechny již dříve probrané druhy grafů

Bitmapové obrázky

frame2im - převod dníkmu filmu na indexovaný obrázek

[X,Map]=frame2in(F) X- indexový obrázek, barva Map

- image** - zobrazí objekt obrázku
image(C) C- je matice, jejíž prvky reprezentují barvu obdelníkového segmentu obrázku
image(x,y,C) - obrázek rozměru x,y
- imagesc** - seřazení dat zobrazení obrázku
- im2frame** - převod obrázku na snímek filmů
- im2java** - převod obrázků na obrázkový objekt Javy
- ind2rgb** - převod indexového obrázků na RGB obrázek

Příklady

- 1) I=imread('obrazek.tif');
javaobrazek=im2java(I)
informace=iminfo(soubor)
- 2) info=iminfo('canoe.tif')
A=imread(soubor)
[A,paleta]=imread(soubor)
imwrite(A,soubor)
- 3) load clown
clims = [10 60];
imagesc(X,clims)
colormap(gray)

1.7.5 Obecná teorie o animaci

MATLAB umožňuje uložit posloupnost snímků (grafických oken) pomocí funkce „getframe“ a následovně je zobrazit zpět pomocí funkce „movie“.

Funkce „**getframe**“ vrací obraz snímku v pixelech. Každý snímek obsahuje binární data uložená ve sloupcových vektorech. Obsah snímku nemá vliv na délku požadovaného sloupce, ale velikost snímku ano. Větší snímek vyžaduje více prostoru!

Funkce „**moviein(n)**“ vytvoří dostatečně velkou matici pro uchování „n“ sloupců, kde každý sloupec obsahuje jeden snímek. Tento krok není nutný, ale pokud není vytvořena matice „M“, kódování probíhá mnohem pomaleji, protože prostor pro „M“ je vždy znovu přiřazován, když se objeví nový sloupec. Pokud byla již animace jednou generována, můžeme ji kdykoliv přehrát.

2 PRAKTICKÁ ČÁST

Program „Elektronická učební pomůcka pro předmět počítačová podpora automatického řízení“ je vytvořen v programu MATLAB a Simulink. Skládá se z jednotlivých souborů typu „*M-file*“. M-file je textový soubor (sekvence příkazů), do něhož lze uložit program, který je dále možno spustit z prostředí MATLAB zadáním jeho přesného názvu do příkazové řádky nebo lze soubor použít v dalším programu a „volat“ ho pomocí příkazu „*callback*“ následovně:

Je vytvořen program, který je uložen pod jménem „*matice*“. Potom se spouští příkazem „*matice*“ v příkazové řádce MATLABu nebo se lze na něj odkázat v dalším programu funkcí „*callback 'matice'*“.

Spuštění programu se realizuje v prostředí programu MATLAB, a to zadáním příkazu „*main*“ na příkazový řádek.

2.1 Panely nástrojů (roletové menu) programu

Po spuštění programu se v hlavním programovém okně (viz. obr. 1) nachází panely nástrojů (Soubor, Menu, Nápověda).

Jednotlivé kapitoly a jejich programový rozvoj je popsán v kapitole 2.2.

Mezi jednotlivými okny se pohybujeme pomocí „push buttons (PB)“ tlačítek a každé okno vytvořené v programu lze tlačítkem „Zpět“ zavřít a vrátit se tak k oknu předchozímu.

Panel nástrojů s označením „*Soubor*“ umožňuje tisk okna „*Tisk*“, ve kterém je program spuštěn nebo ukončení programu pomocí „*Zavřít*“, i programu MATLAB ukončit stisknutím „*Ukončit program*“.

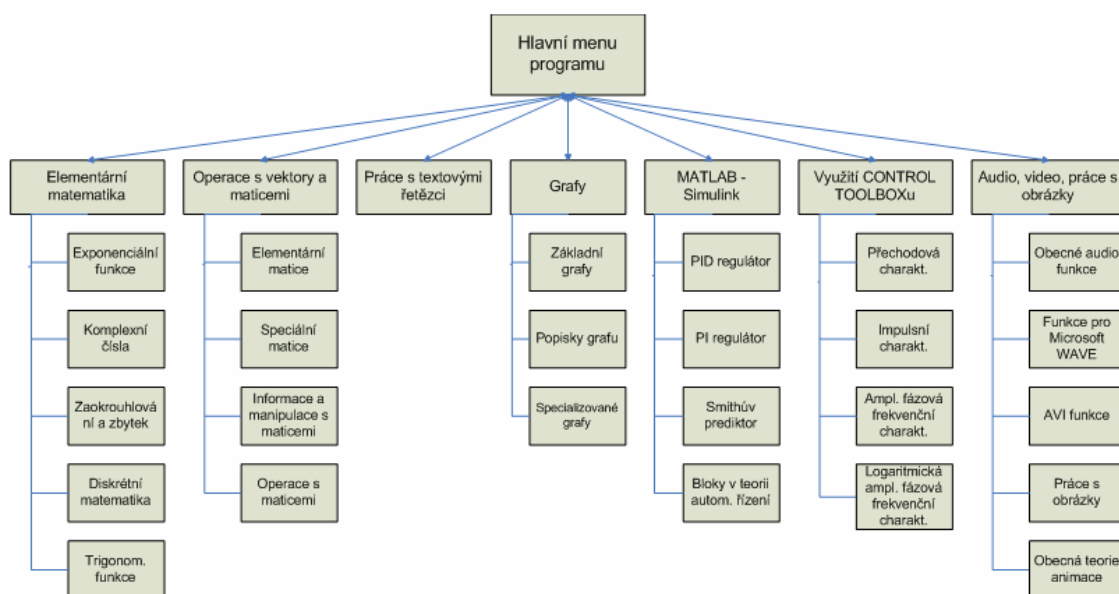
Panel nástrojů „*Menu*“ slouží ke spuštění jednotlivých kapitol. Tedy možnost „*Elementární matematika*“ spustí kapitolu věnující se demonstračním příkladům, taktéž i možnosti „*Operace s vektory a maticemi*“, „*Práce s textovými řetězci*“, „*Grafy*“, „*Jednoduché úlohy pro MATLAB - Simulink*“, „*Využití CONTROL TOOLBOXu*“ a „*Audio, video a práce s obrázky*“.

Panel nástrojů „*Nápověda*“ slouží ke zjištění funkcí, které jsou k programu MATLAB k dispozici (včetně instalovaných toolboxů) a pokud je známo jméno funkce, pak i k určení syntaxe dané funkce. Aktivováním nápovědy se objeví

v prostředí MATLABu celá tato část včetně dostupných funkcí, dále nás tento panel nástrojů pomocí možnosti „O programu“ seznámí s činností a informacemi, k čemu byl program Elektronická učební pomůcka pro předmět počítačová podpora automatického řízení vytvořen.

2.2 Hlavní programové okno

Pro zjednodušení a snadný pohyb v programu slouží následující schéma, které popisuje strukturu celého programu.



Obr. 10 Schéma programu „Elektronická učební pomůcka pro předmět počítačová podpora autom. řízení“

Celá praktická část je vytvořena názorně. V programu jsou použity pouze demonstrační příklady, které ukazují a popisují použité funkce. Parametry jednotlivých funkcí lze měnit podle požadavků uživatele. Je však nutné, aby uživatel měl aspoň základní znalosti programu MATLAB a prostředí Simulink.

V úvodních oknech jednotlivých kapitol se v editačním okně zobrazuje aktuální datum, jehož zdrojový kód je uveden v příloze č.2.

ZÁVĚR

Tato bakalářská práce byla vytvořena za účelem zjednodušení studia programu MATLAB a prostředí Simulink. Cílem bylo vytvořit program, který by poskytoval přehled o práci v prostředí MATLAB. Výsledkem práce je program s názvem: „Elektronická učební pomůcka pro předmět počítačová podpora automatického řízení“. Tento program je vytvořen v programu MATLAB a jeho nadstavbě Simulink. Poskytuje přehled frekventovaných funkcí, které jsou zaměřeny také na oblast automatizace a řídicí techniky, z čehož vyplývá, že uživatel by měl mít základní znalosti této problematiky. Program je rozdělen do několika částí (kapitol). Všechny kapitoly jsou řešeny a demonstrovány tak, aby umožnily snadnou obsluhu programu a ověření získaných vědomostí.

Struktura programu je vytvořena tak, že většina příkladů nemá pevně zadané hodnoty jednotlivých proměnných a funkcí. Uživatel tudíž může jednotlivé funkce kdykoliv odzkoušet. Uveďme např. kapitolu „Operace s vektory a maticemi“, kde uživatel může činnost jednotlivých funkcí přímo vyzkoušet a to zadáním jednotlivých parametrů. Výsledek se vždy zobrazí v editačním okně. Dále jsou zde zastoupeny i takové příklady, ve kterých si uživatel může definovat své hodnoty (například hodnoty přenosu v kapitole „Využití CONTROL TOOLBOXu“) a nechat si programem vypočítat požadovanou funkci nebo zobrazit možnou charakteristiku přenosu. Pro příklad ještě uveďme kapitolu s názvem „Grafy“, kde všechny zmíněné grafy lze zobrazit pomocí tlačítka „Ukázky grafů“ v novém okně a následně nechat vykreslit.

Každá kapitola a příklad obsahují teoretickou část a nápovědu. Po stisku příslušného tlačítka s názvem „Teorie“ nebo „Nápověda“ se otevře nové okno s textem, který většinou popisuje teorii k dané kapitole nebo příkladu.

Program „Elektronická učební pomůcka pro předmět počítačová podpora automatického řízení“ je tedy možno využít jako studijní (demonstrační) materiál pro předmět programová podpora automatického řízení.

SEZNAM POUŽITÉ LITERATURY

- [1] MATLAB a Simulink, Úvod do používání, Ing. František Dušek, Csc., Univerzita Pardubice 2000.
- [2] MATLAB pro Windows, díl II. – popis funkcí, Blanka Heringová, Petr Hora, H-S 1995.
- [3] MATLAB pro začátečníky, Zaplatílek, K.Doňar, Technická literatura BEN,2003.
- [4] Building a Grafical User Interference. Reference Guide. Math Works Inc.
- [5] MATLAB – Základy pro studenty informačních technologií a automatizace, Karel Perůtka, Univerzita Tomáše Bati ve Zlíně 2004.

Internetové stránky : <http://control.zcu.cz/> - Katedra kybernetiky – Fakulta aplikovaných věd, Západočeská univerzita v Plzni.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

I/O	Input/Output (vstup/výstup)
MŘS	Model řízeného systému
KMŘS	Kompenzovaný model řízeného systému
AVI	Audio Video Interleaved
WAV	Formát hudebního souboru
PB	Push buttons
ANS	The most recent answer (odpověď)
D	Den
M	Měsíc
R	Rok
PI	Ratio of a circle's circumference to its diameter (Ludolfovo číslo)

SEZNAM OBRÁZKŮ

Obr. 1 Menu programu „Elektronická učební pomůcka pro předmět počítačová podpora...	9
Obr. 2 Parametry simulace MATLAB Simulink	35
Obr. 3 Princip PI regulátoru.....	37
Obr. 4 Princip Smithova prediktoru, struktura s PI(D) regulátorem	38
Obr. 5 Objekt s jedním vstupem a jedním výstupem.....	38
Obr. 6 Měření přechodové charakteristiky	39
Obr. 7 Diracův impuls	40
Obr. 8 Impulsní charakteristika	40
Obr. 9 Frekvenční charakteristika. a) záznam vstupu a výstupu systému. b) charakteristika v komplexní rovině	41
Obr. 10 Schéma programu „Elektronická učební pomůcka pro předmět počítačová podpora autom. řízení“	46

SEZNAM TABULEK

Tab. 1 Příkazy pro práci s exponenciálními funkcemi	10
Tab. 2 Příkazy pro práci s komplexními čísly	10
Tab. 3 Příkazy pro práci se zaokrouhlováním a se zbytkem	11
Tab. 4 Příkazy pro diskrétní matematiku.....	11
Tab. 5 Příkazy pro práci s trigonometrickými funkcemi	11
Tab. 6 Příkazy pro tvorbu elementárních matic.....	15
Tab. 7 Příkazy pro tvorbu speciálních matic	15
Tab. 8 Příkazy pro práci s informacemi a manipulacemi matic	15
Tab. 9 Příkazy pro operaci s maticemi	16
Tab. 10 Příkazy pro práci se znaky a řetězci	17

SEZNAM PŘÍLOH

P 1 : CD-ROM:	53
P 2 : Datum – zdrojový kód:	54

P 1 : CD-ROM:

Příloha obsahuje program „Elektronická učební pomůcka pro předmět počítačová podpora automatického řízení“, kompletní zdrojové kódy jednotlivých oken programu a elektronickou podobu bakalářské práce.

P 2 : Datum – zdrojový kód:

V bakalářské práci se vyskytuje textové okno s aktuálním datem. Pro jeho zobrazení byl použit následující zdrojový kód:

```
c=clock;
a=zeros(1,3);
a(1)=c(3);
a(2)=c(2);
a(3)=c(1);

cas= uicontrol('BackgroundColor',[1 1 0],'style','text','string','', 'position',[470 360
140 18]);

if a(2)==1
    mesic='-Leden-'
elseif a(2)==2
    mesic='-Únor-'
elseif a(2)==3
    mesic='-Březen-'
elseif a(2)==4
    mesic='-Duben-'
elseif a(2)==5
    mesic='-Květen-'
elseif a(2)==6
    mesic='-Červen-'
elseif a(2)==7
    mesic='-Červenec-'
elseif a(2)==8
    mesic='-Srpen-'
elseif a(2)==9
    mesic='-Září-'
elseif a(2)==10
    mesic='-Říjen-'
elseif a(2)==11
    mesic='-Listopad-'
elseif a(2)==12
    mesic='-Prosinec-'
end;
str=num2str(a(1));
str=strcat(str,mesic);
str=strcat(str,' ');
str=strcat(str,num2str(a(3)))
set(cas,'String',str);

text=uicontrol(figure1,...
'BackgroundColor',[1 1 0],...
'Units','characters',...
```

```
'Style','text',...  
'Position',[93.8 29.2 28 1.5],...  
'Tag','text',...  
'FontWeight','bold',...  
'String','Datum:');
```