

**Studijní podklady k předmětu PPAŘ:
Základy práce s programem MATLAB a
nástroji GUIDE a Profiler**

Supporting evidence for PPAŘ subject:
Basics of MATLAB, GUIDE and Profiler

Martin ADOLF

Bakalářská práce
2005



Univerzita Tomáše Bati ve Zlíně
Fakulta technologická

Institut řízení procesů a aplikované informatiky

Vložit oficiální zadání bakalářské práce

Chtěl bych tímto poděkovat především vedoucímu bakalářské práce panu Ing. Karlu Perůtkovi za pomoc, kterou mi poskytoval během průběhu řešení bakalářské práce.

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

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

Ve Zlíně, 21. 05. 2005

.....

Martin ADOLF

Anotace

Tato práce pojednává o textu, který je zaměřen na základy práce v programu MATLAB. Snaží se o jednoduchý, ucelený a názorný výklad s rozšířením o problematiku dalších nástrojů MATLABu. Je zde podrobně probrána funkce grafického prostředí GUIDE a také tvorba ukázkového programu v tomto prostředí.

Annotation

This thesis deals with text documentation, which includes first steps and basics of MATLAB. It tries to provide simple, compact and illustrative explanation of it with extension to the area of other important tools of MATLAB, namely there is examined in detail the function of GUIDE graphical environment. The description of GUIDE is supported by a demonstrative program created in this environment.

OBSAH

ÚVOD.....	6
1 MATLAB.....	7
1.1 Co je MATLAB?.....	7
1.2 Struktura programu MATLAB	9
1.3 Základní příkazy MATLABu	10
1.4 Základní matematické operace a funkce MATLABu	12
1.4.1 Práce s komplexními čísly	14
1.5 Formátování, práce se vstupem a výstupem	15
1.5.1 Zápis funkce	16
1.5.2 M-Soubory.....	18
1.6 Práce s vektory a maticemi	18
1.7 Základní práce s grafickým výstupem.....	20
1.7.1 Příkaz SUBPLOT	20
1.8 Návod v MATLABu	21
2 GUIDE – NÁSTROJ PRO INTERAKTIVNÍ TVORBU GRAFICKÉHO ROZHRAŇÍ.....	22
2.1 Základní atributy tvorby grafického uživatelského rozhraní (GUI)	22
2.2 Prostředí GUIDE	23
2.3 Tvorba grafických objektů pomocí GUIDE.....	30
2.3.1 Slider (posuvník)	30
2.3.2 Edit Text (přepisovatelné textové pole).....	32
2.3.3 Přehled jednotlivých Uicontrol objektů a jejich zpětnovazební kódy (Callback)	32
2.3.4 Zarovnávání objektů	37
3 UKÁZKOVÝ PROGRAM POMOCÍ GUIDE MATLABU	38
4 NÁSTROJ PROFILER	40
ZÁVĚR.....	41
SEZNAM POUŽITÉ LITERATURY	42
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	43
PŘÍLOHY	44
SEZNAM OBRÁZKŮ.....	50
SEZNAM TABULEK	51
SEZNAM PŘÍLOH.....	52

ÚVOD

Tato práce si klade za cíl vytvořit text pro výuku předmětu, kde je používán programový systém MATLAB. Vytvořený text je zaměřen na základy práce s MATLABem a dále s nástrojem GUIDE a Profiler.

V překotně se vyvíjejícím světě výpočetní techniky a programového vybavení mají od prvopočátku nezastupitelně 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 asi do tří oblastí – tabulkové procesory s rozšířenými matematickými funkcemi, balíky specializované na řešení určitého okruhu problémů (statistika, kvantová chemie, chemické inženýrství atd.) a profesionální univerzální matematické balíky. Do poslední jmenované oblasti patří mj. MathCad, Mathematica a MATLAB¹.

MATLAB je integrované prostředí pro vědeckotechnické výpočty, modelování, návrhy algoritmů, simulace, analýzu a prezentaci dat, měření a zpracování signálů, návrhy řídicích a komunikačních systémů. MATLAB je nástroj, jak pro pohodlnou interaktivní práci, tak pro vývoj širokého spektra aplikací.

Výhoda MATLABu je nejen v jeho velkých možnostech, ale i v tom, jak je široce rozšířen v průmyslu a jeho verze existují pro řadu operačních systémů (Unix, Linux, Windows, Open VMS, IRIX, Solaris, Macintosh, HP-UX a další). MATLAB je komerční software, ale existuje i jeho GNU varianta Octave. Není to samozřejmě to samé, ale syntaxe příkazů a práce s tímto produktem je stejná jako v MATLABu, chybí další nadstavby a vylepšení, ale pro samotné výpočty je to velmi zajímavý nástroj. Hlavně je zdarma a tak přístupnější studentům i pro domácí použití. Studenti mají možnost také výhodně si pořídit studentskou verzi MATLABUu, kde je rovněž velmi výhodná cena proti komerční verzi [9].

¹ MathCad® je chráněná značka fy MathSoft Inc.
Mathematica® je chráněná značka fy Wolfram Software, Inc.
MATLAB® je chráněná značka fy The MathWorks, Inc.

1 MATLAB

1.1 Co je MATLAB?

MATLAB[®] je integrované prostředí pro vědeckotechnické výpočty, modelování, návrhy algoritmů, simulace, analýzu a prezentaci dat, měření a zpracování signálů, návrhy řídicích a komunikačních systémů. MATLAB je nástroj jak pro pohodlnou interaktivní práci, tak pro vývoj širokého spektra aplikací. Typické použití zahrnuje:

- Matematické výpočty
- Tvorba algoritmů
- Získávání dat
- Modelování a simulace
- Analýza dat a vizualizace
- Vědecká a inženýrská grafika
- Tvorba aplikací, včetně grafického rozhraní
- Propojení s tabulkovým editorem Excel

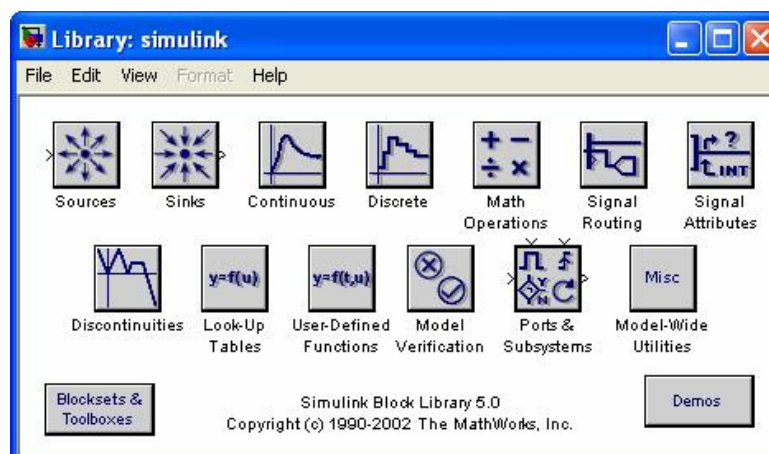
Program procházel řadu let dlouhým vývojem. Obsahuje nástroje pro tvorbu a ladění programů, analýzu vytvořeného kódu a také nástroje pro publikování zdrojových kódů ve formě HTML nebo MS-Word. Důležitým rozšířením je interaktivní nástroj pro tvorbu grafů GUIDE, který nejen usnadňuje a zrychluje práci s grafikou, ale je schopen automaticky generovat zdrojový kód grafiky – ten je možné následně použít pro další vlastní aplikace nebo distribuovat jiným uživatelům [1, 2, 3, 6].

Ve srovnání s jinými obdobnými produkty je MATLAB blíže klasickému programovacímu jazyku, jako např. C. Nemá tolik hotových komplexních matematických funkcí jako např. Mathematica, nemá vlastnosti textového editoru jako např. MathCad. Zrovna tak není jeho standardní součástí podpora symbolických výpočtů jakou obou výše jmenovaných produktů.

Ale to neznamená, že je o tyto možnosti ochuzen. Základní instalace obsahuje velké množství (více než 500) jednoduchých² i složitějších³ matematických funkcí implementovaných ve formě vysoce efektivních a robustních algoritmů, které jsou součástí jádra (built-in function). Z těchto funkcí lze složit podle konkrétní aplikace libovolně složité další funkce [6].

Skupiny takovýchto funkcí respektive funkcí hodících se k řešení určitého okruhu problémů se v MATLABu nazývají toolboxy. Obvykle byly vytvořeny (nebo aspoň jejich základ) na univerzitách týmem seskupeným okolo významného odborníka v dané oblasti.

Jako samostatná nadstavba MATLABu existuje SIMULINK, který slouží k řešení diferenciálních rovnic s grafickým zadáváním řešené soustavy připomínajícím zapojení na analogovém počítači. Umožňuje graficky sledovat průběhy veličin v libovolném místě zapojení. Používá se tedy například pro simulaci dynamického chování sledovaného systému.



Obr. 1.1 Knihovna toolboxů Simulink

² všechny běžné matematické funkce (pracující i s komplexními hodnotami) a základní operace s maticemi

³ např. Besselovy funkce, gama funkce, generování prvočísel, transformace souřadnic, Fourierova transformace, maticové funkce, funkce pro práci s řídkými maticemi, funkce pro práci s polynomy, numerická integrace a derivace funkce, nulová hodnota funkce, minimum funkce více proměnných, řešení soustavy diferenciálních rovnic I. řádu atd.

Simulink[®] je program pro simulaci a modelování dynamických systémů, který využívá algoritmy MATLABu pro numerické řešení nelineárních diferenciálních rovnic. Poskytuje uživateli možnost rychle a snadno vytvářet modely dynamických soustav ve formě blokových schémat a rovnic. Nový přístup k řešení diferenciálních rovnic dovoluje simulovat i rozsáhlé „stiff“ systémy rychle, přesně a s efektivním využitím paměti počítače.

Pomocí Simulinku a jeho grafického editoru lze vytvářet modely lineárních, nelineárních, v čase diskrétních nebo spojitých systémů pouhým přesouváním funkčních bloků myší. Simulink nově umožňuje spouštět určité části simulačního schéma na základě výsledku logické podmínky. Tyto spouštěné a povolované subsystémy umožňují použití programu v náročných simulačních experimentech. Samozřejmostí je otevřená architektura, která dovoluje uživateli vytvářet si vlastní funkční bloky a rozšiřovat již tak bohatou knihovnu Simulinku. Hierarchická struktura modelů umožňuje koncipovat i velmi složité systémy do přehledné soustavy subsystémů prakticky bez omezení počtu bloků. Simulink, stejně jako MATLAB, dovoluje připojovat funkce napsané uživateli v jazyce C. Vynikající grafické možnosti Simulinku je možné přímo využít k tvorbě dokumentace. Mezi neocenitelné vlastnosti Simulinku patří nezávislost uživatelského rozhraní na počítačové platformě. Přenositelnost modelů a schémat mezi různými typy počítačů umožňuje vytvářet rozsáhlé modely, které vyžadují spolupráci většího kolektivu řešitelů na různých úrovních [7, 8].

1.2 Struktura programu MATLAB

Strukturu systému MATLAB lze rozdělit na 5 hlavních částí:

Vývojové prostředí. Tato sada nástrojů a prostředků pomáhá v práci s funkcemi a soubory. Mnoho těchto nástrojů má grafické uživatelské prostředí. To zahrnuje plochu MATLABu, příkazové okno, historii příkazů, editor, ladící software a prohlížeč helpu, plochy, souborů a vyhledávací cesty.

Knihovna matematických funkcí MATLABu. To je obrovská sada výpočetních algoritmů v rozsahu od elementárních funkcí jako *suma*, *sin*, *cos* a komplexní aritmetika

k více sofistikovaným funkcím jako je *inverze matice*, *vlastní čísla matice*, *Besselovy funkce* a *rychlá Fourierova transformace*.

Jazyk MATLAB. To je pokročilý jazyk pro práci s maticemi a poli s možností řízení výpočtu. Jazyk umožňuje práci s datovými strukturami, s vstupem a výstupem. Je zahrnuta podpora objektového programování. To umožňuje dvě věci, jak velmi rychle udělat jednoduchý program, tak vytvářet složité komplexní aplikace.

Grafika. MATLAB má rozsáhlé prostředky pro zobrazení vektorů a matic jako grafy, právě tak jako možnosti popisu a tisku těchto grafů. MATLAB má mnoho funkcí pro 2D a 3D vizualizaci dat, zpracování obrazu, animací a prezentací grafiky.

Rozhraní MATLABu pro aplikační programy (API). Tato knihovna nám umožňuje psát programy v jazyku C a ve Fortranu, které komunikují s MATLABem. To zahrnuje prostředky pro volání podprogramů z MATLABu (dynamické linky), volání výpočetního jádra MATLABu, a umožňuje čtení a zápis MAT-souborů.

1.3 Základní příkazy MATLABu

- pro přerušení prováděného příkazu lze použít klávesovou zkratku CTRL+C
- klávesy šipka nahoru/dolů umožňují pohyb po historii příkazů
- editace řádku se provádí šipkami vlevo/vpravo, klávesami *delete*, *backspace*
- klávesa ESC vymaže celý řádek
- proměnné v MATLABU musí začínat písmenem a mohou mít až 31 znaků
- rozlišují se malá a velká písmena
- příkaz ukončený středníkem (;) se nevypisuje na obrazovku

close name – uzavře námi požadovanou aktivní „figure“ – graf

close all – uzavře všechna okna kromě „MATLAB Editor/Debugger“

dir – vypíše obsah aktuálního adresáře – obdobně jako DOS

path – vypíše všechny přístupné adresáře

what – vypíše jednotlivé soubory MATLABu podle typu (resp. přípony) z aktuálního adresáře

type – vypíše obsah souboru s příponou *.m, bez otevření *MATLAB Editor /Debugger*

who – vypíše nám na obrazovku veškeré proměnné, které jsou momentálně zadány do MATLABu

whos – stejný výpis jako who, jen s tím rozdílem, že nám MATLAB vypíše nejen názvy proměnných, ale také jejich velikost (*Size*), počet bytů alokovaných pro dané pole dané proměnné (*Bytes*) a jejich typ (*Class*)

workspace – otevře *Workspace Browser*, který nám podává stejné informace jako **whos**. Rozdíl je v tom, že nám umožňuje jednotlivé proměnné pohodlně vymazávat, nebo si tyto proměnné otevřít v *MATLAB Editor/Debuggeru*, který nám zobrazí jejich hodnoty a kde můžeme tyto hodnoty velmi jednoduše přepisovat a měnit samotnou velikost proměnné (prodloužit, zkrátit, přidat sloupce). Pokud v *Workspace Browseru* dvakrát klikneme pomalu na nějakou proměnnou, můžeme změnit také její název.

edit – tento příkaz nám otevře *MATLAB Editor /Debugger*

edit name – nám otevře *MATLAB Editor /Debugger* se zadaným souborem

Pokud chceme smazat nějakou proměnnou nebo pokud chceme smazat všechny proměnné, použijeme příkaz **CLEAR**

clear name – kde *name* představuje název proměnné

clear all – smaže všechny proměnné

clc – vymaže obrazovku v MATLABu (*Clear Command Window*)

clf – pokud máme otevřený nějaký graf a chceme ho vymazat, použijeme příkazu *clf* (*Clear figure*), vymaže aktuální „figure“

echo on/off Pokud nechceme vypisovat při spouštění M-souboru, co nám který řádek provádí, napíšeme na konec středník. Pokud později chceme vypisovat veškeré řádky včetně jejich příkazů, použijeme příkaz **echo on**. Tento příkaz nám začne vypisovat nejen řádky se středníkem na konci, ale také informační řádky (před ty se na začátek řádku dává znak %). Pro vypnutí tohoto zobrazování máme funkci **echo off**.

size – tento příkaz nám slouží ke zjištění velikosti matice, kde první číslo z výsledku znamená počet řádků a druhé číslo počet sloupců.

length – příkaz **LENGTH** nám vrací délku proměnné a to délku která je „větší“, což znamená, že pokud máme matici $M \times N$, kde $M > N$, tak nám vrací hodnotu M . Jinak se dá říci, že příkaz **LENGTH** nám vrací „větší“ hodnotu z příkazu **SIZE**.

input – při programování v M-souborech se používá **INPUT** pro načtení proměnné z klávesnice

- použití: `A = input ('zde napis hodnotu promenne A:')`

disp – při programování v M-souborech se tento příkaz používá, pokud chceme vypsát nějakou hlášku uživateli na obrazovku.

- použití: `disp('Toto je text, který vam chci sdelit.')`

pause (n) – příkaz pro pozdržení dalších výpočtů, kde n představuje čas v sekundách, na který se výpočet zastaví. Pokud napíšeme **PAUSE** – bez udání času, MATLAB bude čekat na stisk libovolné klávesy.

1.4 Základní matematické operace a funkce MATLABu

Při násobení proměnné číslem např. a a 5, se nesmí zapomínat na operaci násobení, jinak program hlásí chybu. Správně psaná syntaxe vypadá tedy $5*a$. Přiřazení číselné hodnoty do proměnné a se provádí velmi jednoduchým způsobem, pomocí znaku $=$, tj. např. $a=\text{číselná hodnota}$ [1, 9].

Tabulka I. Základní matematické operace

Operace	Symbol	Příklad
sčítání, $a+b$	+	$52+65$
odčítání, $a-b$	-	$22-35$
násobení, $a \cdot b$	*	$28 \cdot 15$
dělení, $a:b$	/ nebo \	$58/6=58 \setminus 6$
mocnění, a^b	^	$3^2=9$

MATLAB v sobě obsahuje, několik předem nadefinovaných proměnných, které se využijí při výpočtu. Kromě známých π (π – Ludolfovo číslo) a komplexních jednotek (i nebo j) jsou tam další, které souvisí s numerickou matematikou počítačů (*eps*, *realmin*, *realmax*, *inf*, *NaN*, *flops*). Důležité jsou *inf* (nekonečno) a *Nan* (neplatná numerická operace), se kterými se můžeme při výpočtech častěji setkat. Přetypování speciálních proměnných je možné a jejich hodnota se obnoví po vymazání proměnné se stejným názvem (*clear* nebo *clear proměnná*). Seznam speciálních proměnných uvádí tabulka II.

Tabulka II. Speciální proměnné

Speciální proměnné	Popis
ans	proměnná k uložení výsledku
pi	Ludolfovo číslo
eps	nejmenší použitelné číslo
flops	počet operací
inf	označení pro nekonečno (např. 1/0)
Nan, nan	označení nedefinované hodnoty
i,j	komplexní jednotka
nargin	počet vstupních parametrů funkce
nargout	počet výstupních parametrů funkce
realmin	nejmenší použitelné kladné reálné číslo
realmax	největší použitelné kladné reálné číslo

1.4.1 Práce s komplexními čísly

Pro práci s komplexními čísly slouží funkce v tabulce III. Komplexní číslo se zapisuje ve tvaru $a+j*b$ nebo $a+i*b$.

Tabulka III. Komplexní čísla

Komplexní funkce	Popis
abs	absolutní hodnota nebo velikost
angle	fázový úhel v <i>rad</i>
conj	komplexně sdružené číslo
imag	imaginární část komplexního čísla
real	reálná část komplexního čísla
unwrap	konverze na stejný fázový úhel
isreal	funkce je <i>true</i> pro reálné pole
cplxpair	setřídí vektor podle velikosti reálných částí

Tabulka IV. Exponenciální funkce

Název funkce	Popis
exp	exponenciální funkce
log	logaritmická funkce se základem e
log10	logaritmická funkce se základem 10
log2	logaritmická funkce se základem 2
pow2	druhá mocnina
unwrap	konverze na stejný fázový úhel
sqrt	druhá odmocnina
nextpow2	nejbližší vyšší druhá mocnina

Tabulka V. Transformace souřadnic

Název funkce	Popis
cart2sph	transformace z kartézských do sférických
cart2pol	transformace z kartézských do polárních, cylindrických e
pol2cart	transformace z polárních do kartézských
sph2cart	transformace z sférických do kartézských

Tabulka VI. Ostatní funkce

Název funkce	Popis
factor	rozklad na prvočísla
isprime	je <i>True</i> pro prvočísla <i>e</i>
primes	seznam prvočísel
gcd	největší společný dělitel
lcm	nejmenší společný násobek
rat	aproximace zlomkem
rats	vyjádření zlomkem
perms	permutace
nchoosek	<i>n</i> nad <i>k</i>

1.5 Formátování, práce se vstupem a výstupem

Pro zobrazení čísel používán příkaz *format* (tabulka VII. – ukázka formátování čísla *pi* pro různé druhy formátů). MATLAB si uchovává přesnou reprezentaci čísel v paměti, ale jak je chceme zobrazit a pracovat s nimi je na uživateli.

V pracovní ploše vidíme aktuálně definované proměnné, které si můžeme vypsát také pomocí příkazu *who* (vypíše seznam existujících proměnných) a *whos* (opět seznam proměnných, ale s detaily o proměnných).

Pro psaní kódu v MATLABu jsou možné dva přístupy, použijeme-li první, pak zadáváme příkazy v okně příkazů (*Command Window*). To má nevýhodu v tom, že pokud chceme tento výpočet zopakovat, tak musíme vše napsat znovu. Proto se používá spíše druhý způsob a to je vytváření souborů s příponou *.m (M-soubor). Je to textový soubor, který obsahuje sled příkazů MATLABu.

Tabulka VII. Zobrazení dat

Příkaz	π
format short	3.1416
format long	3.14159265358979
format short e	3.1416e+000
format long e	3.14159265358979e+000
format short g	3.1416
format long g	3.14159265358979
format hex	400921fb54442d18
format bank	3.14
format +	+
format rat	355/133

1.5.1 Zápis funkce

Pokud chceme napsat speciální funkci musíme ji vytvořit jako samostatný soubor a musí mít tuto syntaxi.

```
function [out1, out2,...]=jmeno_fce (in1, in2, ...)
    příkazy
end;
```

Potom funkci můžeme volat:

```
[x1,x2]=jmeno_fce(a,b,c);
[x1,x2]=jmeno_fce(2,5,12);
```

Nový M-soubor vytvoříme jednoduše pomocí menu (*File/New/M-file*). Otevře se okno editoru, kde se dají psát příkazy, které spustíme stiskem klávesy F5, nebo přes menu (*Debug/Run*). Příkazy se provedou a výpis uvidíme v okně příkazů, pokud jsme výstup nepřesměrovali např. do souboru. M-soubor se dá spustit také zadáním jména souboru (bez přípony .m) v okně příkazů. Použijeme *cd* pro aktuální adresář a *dir* pro výpis souborů. Pro výpis obsahu souboru můžeme použít příkaz *type*.

V tabulce VIII. jsou příkazy, které se využijí při psaní skriptů (M-souborů). Pro psaní můžeme využít znalosti z programování příkazy typu *if-then*, cykly a podmínky. Stačí jednoduše v okně příkazů napsat *help if* a vypíše se část helpu s ukázkou syntaxe tohoto příkazu, obdobně můžeme postupovat u ostatních [9, 10].

Tabulka VIII. Příkazy používané pro psaní skriptů

Funkce	Popis
disp(proměnná)	zobrazí proměnnou
echo	on, off ve skriptu výpis příkazů do příkazového okna
input	výzva k vložení hodnoty
keyboard	dočasné použití klávesnice, návrat <i>return</i>
pause	čeká na stisk jakékoliv klávesy
pause(<i>n</i>)	čekat <i>n</i> sekund
waitforbuttonpress	čekat dokud nestisknete tlačítko myši nebo klávesu

Tabulka IX. Příkazy pro manipulaci se soubory

Manipulace se soubory	Popis
cd (nebo) pwd	zobrazí aktuální cestu
<i>p</i> =cd	aktuální cesta v řetězci <i>p</i>
delete soubor.m	vymaže <i>soubor.m</i>
dir (nebo) ls	zobrazí soubory v aktuálním adresáři
<i>d</i> =dir	soubory v aktuální cestě ve struktuře <i>d</i>
exist('cow', 'file')	kontroluje existenci souboru <i>cow.m</i>
exist('dname', 'dir')	kontroluje existenci adresáře <i>dname</i>
<i>p</i> =matlabroot	aktuální cesta k programu Matlab v řetězci <i>p</i>
type cow	vypíše soubor <i>cow.m</i>
what	vypíše soubory *.m
which cow	zobrazí cestu k souboru <i>cow.m</i>

1.5.2 M-Soubory

M-soubory (M-files) jsou textové soubory s koncovkou .m, do kterých se píše funkce a skripty. Spouští se napsáním jména souboru na příkazový řádek a stisknutím klávesy ENTER. Je vhodné volit název funkce stejný jako název souboru. Skript může pracovat s proměnnými v paměti MATLABu a také po skončení běhu skriptu hodnoty proměnných zůstanou v paměti. Skripty nemají vstupní ani návratové hodnoty. Funkce používá vlastní paměťový prostor, může mít vstupní argumenty a vracet hodnoty. Po definici hlavní funkce mohou v m-souboru následovat definice dalších funkcí (tzv. subfunctions). Funkce i skripty obsahují příkazy v jazyku MATLAB.

1.6 Práce s vektory a maticemi

Zkratka MATLAB znamená v překladu maticová laboratoř a proto jsou tu všechna čísla reprezentovaná jako vektory a matice. Dále je zde uveden podrobný popis ke konstrukci a práci s vektory a maticemi.

Tabulka X. Způsob zápisu vektorů

Zápis v Matlabu	Význam
<code>a = [1;2;3;4;5]</code>	sloupcový vektor
<code>c = [1,2,3,4,5]</code>	řádkový vektor
<code>b = [1,2,3,4,5]'</code>	transpozice řádkového vektoru

Tabulka XI. Seznam funkcí pro práci s maticemi

Název	Popis funkce
eye	definuje jednotkovou matici
zeros	definuje nulovou matici
ones	definuje matici jedniček
rand	generuje matici náhodných hodnot
det	determinant matice
inv	inverze matice
'	operace transpozice (apostrof)
.'	transformuje prvek po prvku
sqrtm	maticová mocnina
expm	maticová exponenciála
logm	logaritmus matice
poly	charakteristický polynom
size	rozměry matice
roots	vlastní čísla charakteristického polynomu
max	maximální prvek matice
*	maticové násobení
.*	násobení prvek po prvku
^	maticové mocnění
.^	mocnění prvek po prvku
\,/	dělení zleva, zprava
.\	dělení zleva prvek po prvku
./	dělení zprava prvek po prvku
kron	Kronerovo tenzorové násobení

1.7 Základní práce s grafickým výstupem

Naše vypočtené hodnoty či simulované průběhy potřebujeme určitým způsobem prezentovat. Z tohoto důvodu obsahuje program MATLAB funkce sloužící k vykreslení zadaných parametrů. Nejdůležitější z používaných funkcí je funkce `plot`. Funkce `plot` slouží k zobrazení dvou vektorů. Tyto vektory musí mít shodnou dimenzi. MATLAB neumí pouze vykreslovat funkce ve 2D, ale také ve 3D, v logaritmických či polárních souřadnicích. Pro provedení grafického výstupu MATLAB vytvoří tzv. `FIGURE`. Každý *figure* má svoje číslo. Proto pokud si nechceme překreslit náš graf, musíme pro následující příkaz `plot` změnit *figure*. Příkaz `figure` (číslo *figure*). Pro vykreslování více grafů do jednoho `Figure` se používá funkce `subplot` (viz podkapitola 1.7.1).

plot – jednoduchý 2D graf

plot3 – jednoduchý 3D graf

loglog – graf s log. souřadnicemi

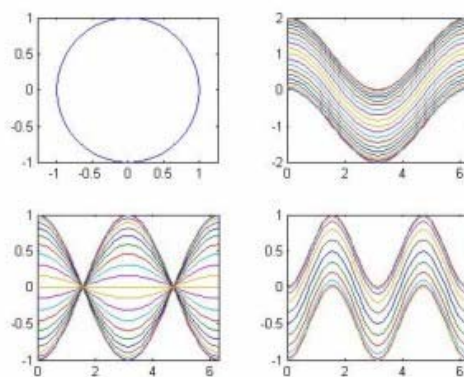
semilogx – graf, kde *x* má log. souřadnice

semilogy – graf, kde *y* má log. souřadnice

plotyy – graf s hlavní a vedlejší osou *y*

1.7.1 Příkaz SUBPLOT

Příkaz `subplot` tiskne do jednoho *figure* více grafů. Parametry jsou `subplot(m,n,p)`, kde *m* je počet grafů v řádku, *n* je počet řádků a *p* je číslo pozice. Tedy pro *m,n* = 2 může být *p* = 1..4. `Subplot` však pouze určuje na jaké místo bude `plot` následně vykreslovat. Sekvence proto musí být `subplot( )`, `plot( )`.



Obr. 1.2 Zobrazení sekvence grafů

1.8 Náповěda v MATLABu

Práce s nápovědou je velmi důležitá, nápověda v MATLABu je buď přístupná pomocí příkazové okna (*Command Window*) nebo si můžeme vyvolat samostatné okno nápovědy. Manuály jsou také na instalačním disku CD-ROM MATLABu uloženy také ve formátu pdf, který je vhodný pro tisk.

Příkazy týkající se nápovědy:

doc – výpis nápovědy do okna nápovědy *Help*

help – nápověda, výpis příkazů a funkcí do okna *Command Window*

help příkaz – nápověda konkrétního příkazu

helpwin – nápověda v okně

lookfor – hledá klíčové slovo v nápovědě

info – informace o MATLABu a firmě Mathworks

2 GUIDE – NÁSTROJ PRO INTERAKTIVNÍ TVORBU GRAFICKÉHO ROZHRAŇÍ

GUIDE, **MATLAB®** je grafické uživatelské vývojové prostředí, poskytující soupravu nástrojů pro vytváření grafických uživatelských rozhraní. V angličtině se pro toto prostředí vžila zkratka **GUI** (Graphical User Interface). Uživatel si také může vytvářet grafické objekty sám přímo v editoru zdrojových textů.

V některých případech, jde zejména o začátečníky či příležitostné uživatele, avšak programátor dá přednost nástroji, který za něho všechny grafické prvky vytvoří. Takový nástroj, jakýsi automatický generátor GUI, je v systému **MATLAB** integrován. Nazývá se **GUIDE** (Graphical User Interface Development Environment). Po jeho spuštění nás tento **GUIDE** vede interaktivním způsobem a my si vybíráme grafické objekty, jako jsou panely, tlačítka, textová pole, posuvníky, menu, atd., myši je umísťujeme do navržené oblasti a zadáváme jejich parametry. Jde o řešení univerzální a časově nenáročné [2, 4].

2.1 Základní atributy tvorby grafického uživatelského rozhraní (GUI)

Při tvorbě grafického uživatelského rozhraní je třeba mít stále na mysli základní poslání GUI – ulehčení ovládání aplikace uživatelem. To vyžaduje, aby byly splněny během celého běhu programu základní požadavky:

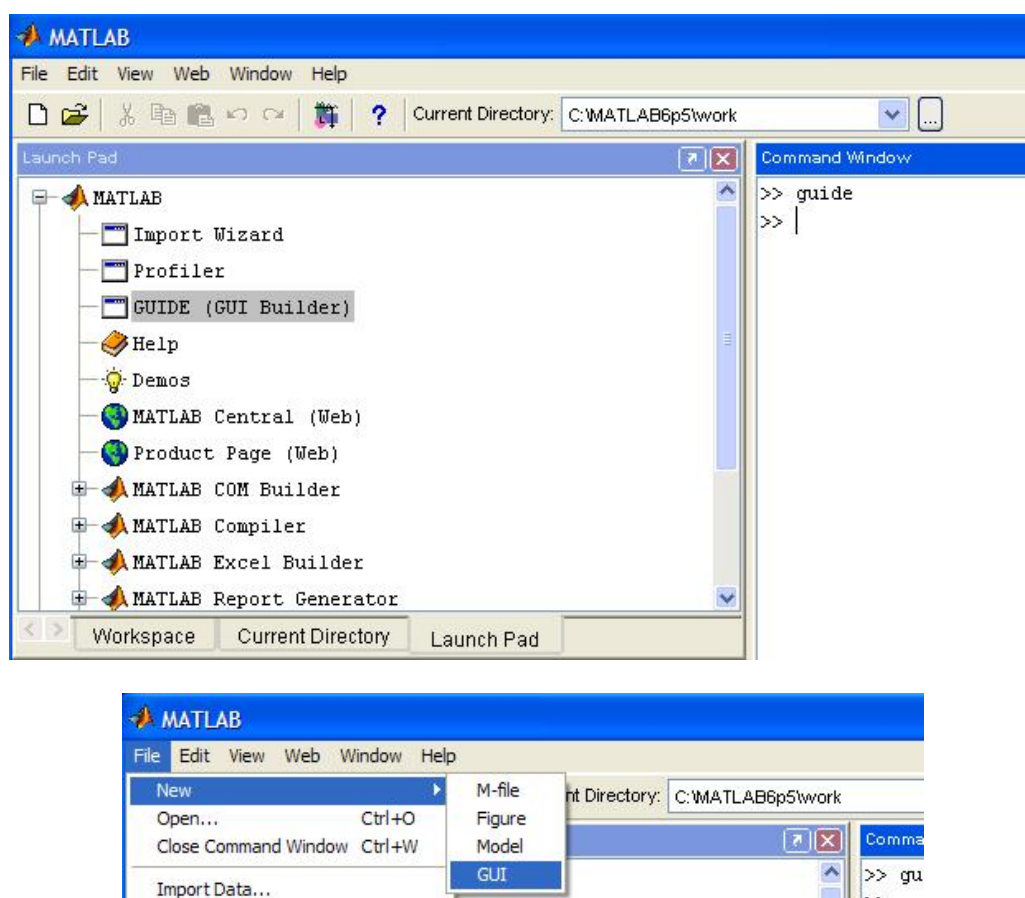
- **JEDNODUCHOST** (robustnost),
pohyb v naší aplikaci by měl být jednoduchý a rychlý. Kliknutí na ikonku, tlačítko či přepínač je vždy rychlé a snadné.
- **PROVÁZANOST** (vždy musí být zajištěn návrat),
uživatel (ať už jsme to někdy později my sami, nebo někdo zcela jiný) naší aplikace se nesmí při řízení jejího běhu ztratit ve slepé uličce bez návratu, tedy:
 - uživateli by mělo být jasné, kde se právě nachází,
 - uživatel by měl vědět jaký je další krok.
- **KOMPLEXNOST** (ošetření všech eventualit).

Může se uživatel vrátit na začátek?

2.2 Prostředí GUIDE

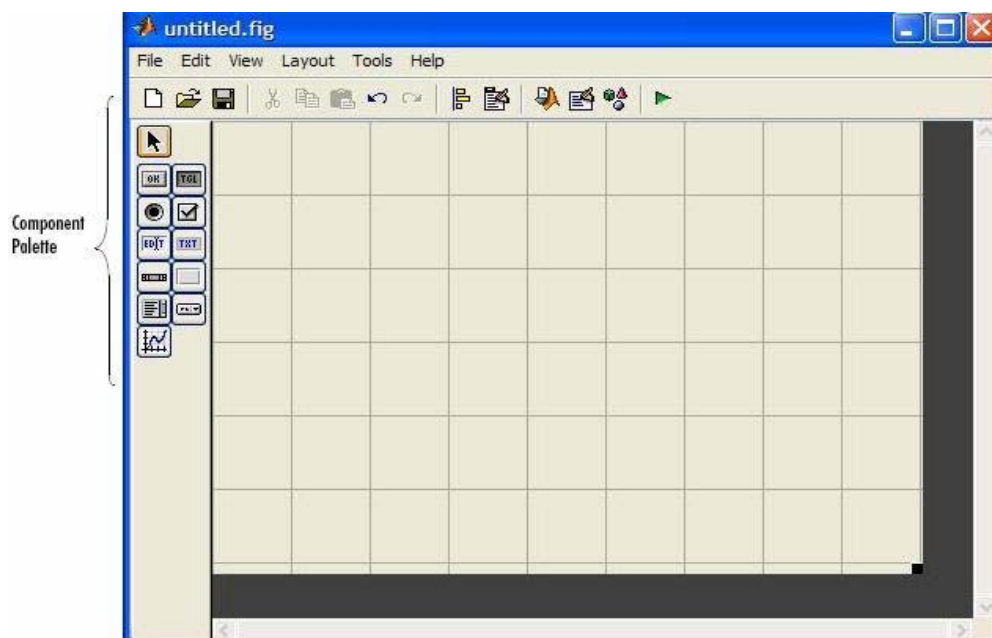
Průvodce tvorbou GUI (GUIDE) je grafické interaktivní vývojové prostředí, obsahující všechny grafické objekty typu **uicontrol**, kterými je možno ovládat běh aplikace. Generuje automaticky základní zdrojový kód pro ovládání. Vzhled vytvořeného GUI ukládá GUIDE do souboru s příponou *.fig a jeho zdrojový kód do M-souboru s příponou *.m.

Vyvolání průvodce je možné více způsoby, které jsou ukázány na obr. 2.1. Jedním z nich je spuštění dvojklikem levého tlačítka myši z okna *Launch Pad*, nebo zápisem a potvrzením příkazu **guide** v hlavním prostředí MATLABu (*Command Window*). Třetí možnost je využít základní menu *File/New/GUI*.



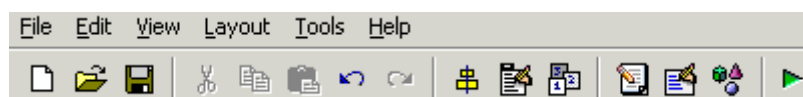
Obr. 2.1 Tři způsoby vyvolání GUIDE

Průvodce má podobu standardního okna s řadou rozbalovacích menu, ze kterých jsou přístupné všechny možnosti a nastavení, a pruhu ikon, které jako obvykle zastupují některé důležité a často používané povely.



Obr. 2.2 Základní vzhled Průvodce tvorbou GUI (GUIDE) o jeho spuštění

Dále na levé straně, v tzv. *Component Palette*, jsou všechny objekty *uicontrol*. Pracovní plochu tvoří čtvercová síť – *Layout Area*, umožňující hrubě vizuálně určovat souřadnice jednotlivých objektů umísťovaných na plochu. V pravém dolním rohu čtvercové sítě je malý černý čtvereček. Ten dovoluje pomocí levého tlačítka myši (kliknutím a držením) přizpůsobit rozměry čtvercové sítě *Layout Area* dle našich potřeb.



Obr. 2.3 Tlačítka (ikony) v horní části otevřeného okna GUIDE

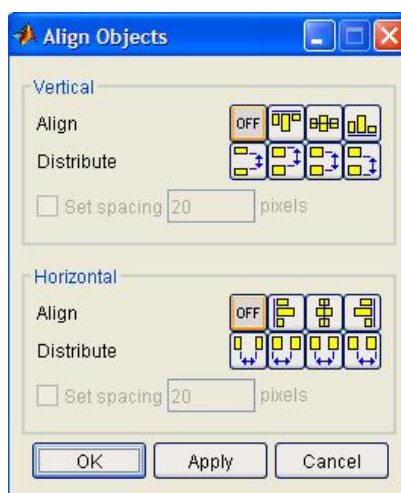
První dvě vodorovné sekce ikon (obsahující 3 a 5 položek) jsou standardní a dobře známé, další tři sekce jsou specifické, obr. 2.3. Najetím myši na jednotlivé ikonky se standardně ve žlutém políčku zobrazí jejich funkce. Všechny tyto nové ikonky jsou vybarvené a tudíž aktivní.

Devátá ikona zleva (první v třetí sekci) je *Alignment Tool* – nástroj pro zarovnávání grafických objektů na:

- horní nebo levý okraj,
- na středy,
- na spodní nebo pravý okraj,

nebo jejich rozložení na všechny možné způsoby:

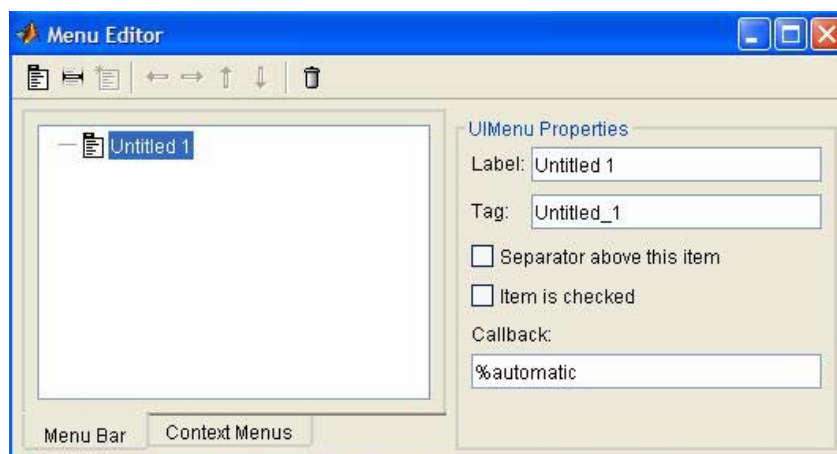
- pevná vzdálenost mezi objekty,
- pevná vzdálenost mezi horními nebo levými okraji,
- pevná vzdálenost mezi středy objektů,
- pevná vzdálenost mezi spodními nebo pravými okraji.



Obr. 2.4 Možnosti použití nástroje Alignment Tool

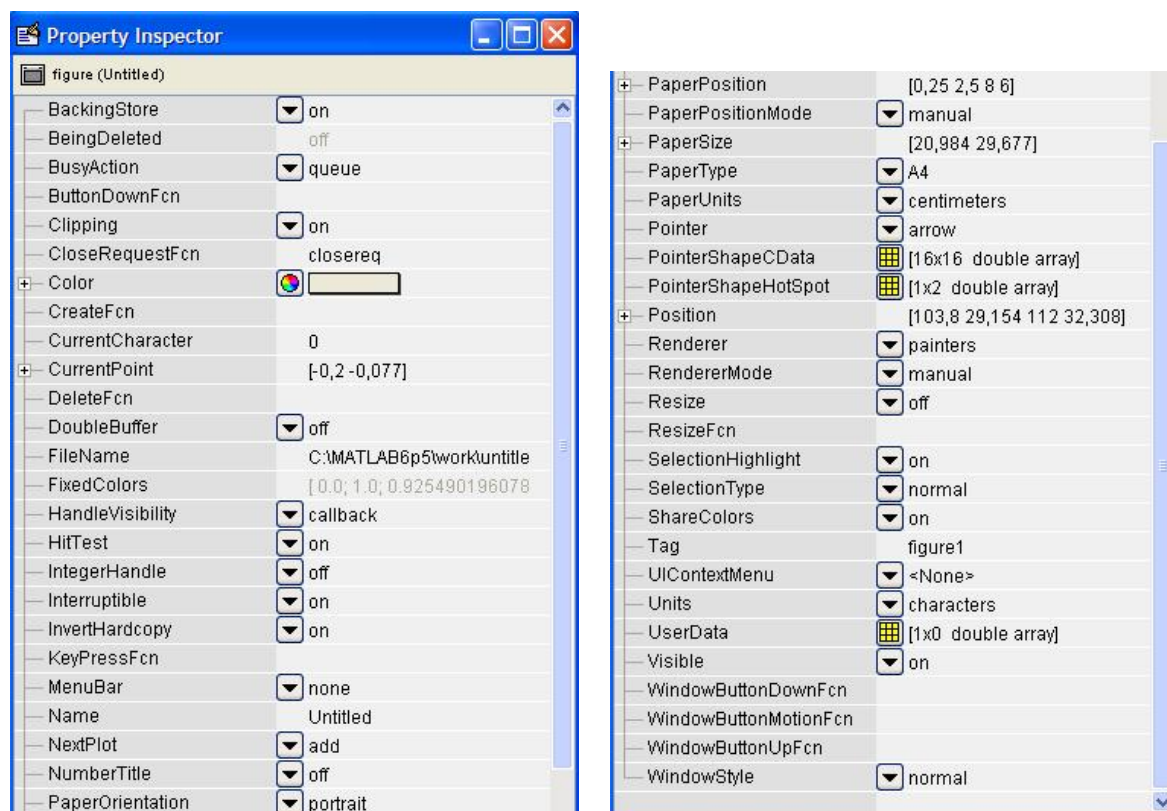
Desátou ikonou je *Menu Editor* (česky asi netřeba), obr. 2.5.

Po kliknutí na ikonku se otevře okno bez pravé strany. Až po kliknutí na položku na bílé ploše (zde *Untitled1*) se oživí i pravá strana, jak ukazuje obrázek. Už teď jsou ze záložek levého okénka zřejmé dva druhy menu – hlavní a kontextové.



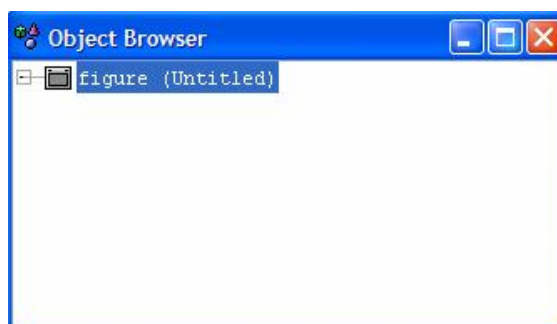
Obr. 2.5 Spuštění Menu Editoru

Jedenáctou ikonou (první ve čtvrté sekci) spustíme tzv. *Property Inspector* okno pro řízení vlastností, obr. 2.6. Jsou tam opravdu všechny a mnohé z nich použijeme jen výjimečně. *Property Inspector* můžeme vyvolat i dvojklikem přímo v *Layout Area*.

Obr. 2.6 Okno tzv. *Property Inspectoru* – okno pro řízení vlastností GUI

Kliknutím na dvanáctou ikonku spustíte tzv. *Object Browser* – prohlížeč objektů, obr. 2.7. Tam se nám zobrazí všechny použité objekty celé aplikace a jejich hierarchie.

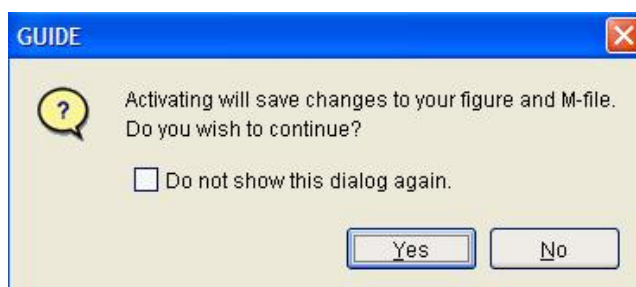
Dvojklikem levým tlačítkem myši na *figure (Untitled)* v *Object Browseru* vyvoláme opět *Property Inspector*.



Obr. 2.7 Spuštění prohlížeče objektů (*Object Browser*)

Poslední ikonka v podobě zelené trojúhelníkové šipky má funkci tzv. *Activate figure* obr. 2.8. Uloží všechny změny ve vzhledu GUI do souboru obrázku *.fig a uloží je též do příslušného M-souboru *.m Máme možnost říci „ano“ nebo „ne“. Anebo v jediném zaškrtnutém políčku „Tento dialog v budoucnu nezobrazovat“, sdělíme trvalé „ano“.

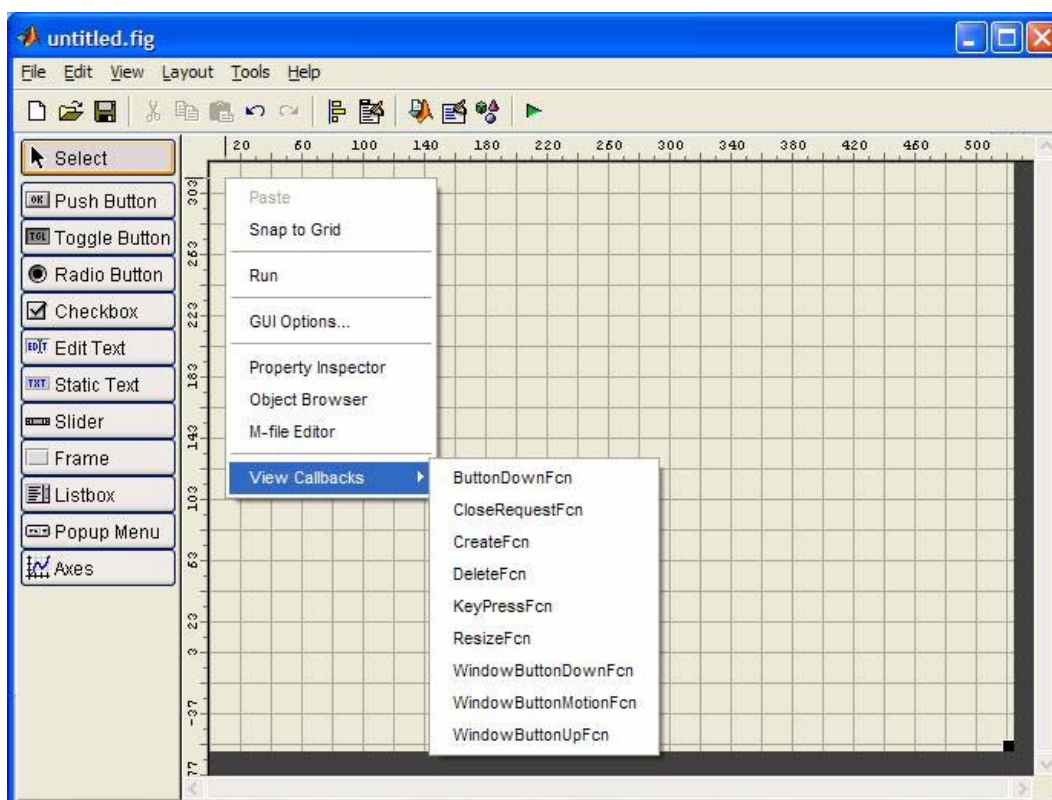
V tuto chvíli, pokud povolíte uložení, bude založen soubor např. untitled1.fig a untitled.m v příslušné aktuální složce (např. Work).



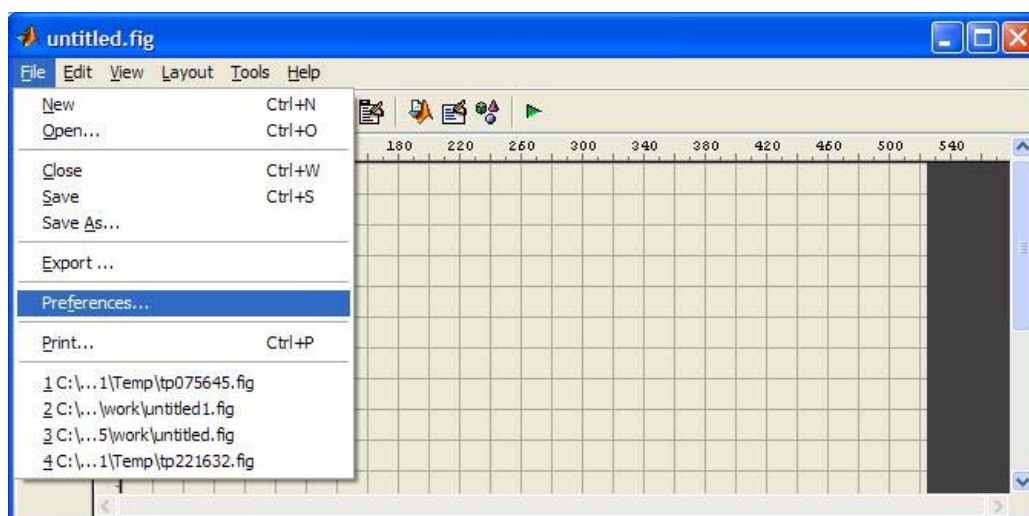
Obr. 2.8 Uložení vzhledu GUI

V tuto chvíli stojí ještě za zmínku z rozbalovacích menu dvě položky. Jednou z nich je položka *File/Preferences*.

Po zvolení položky *Preference* se otevře okno zobrazené na obr. 2.11. Zde si můžeme nastavit prostředí systému MATLAB podle našich aktuálních potřeb.



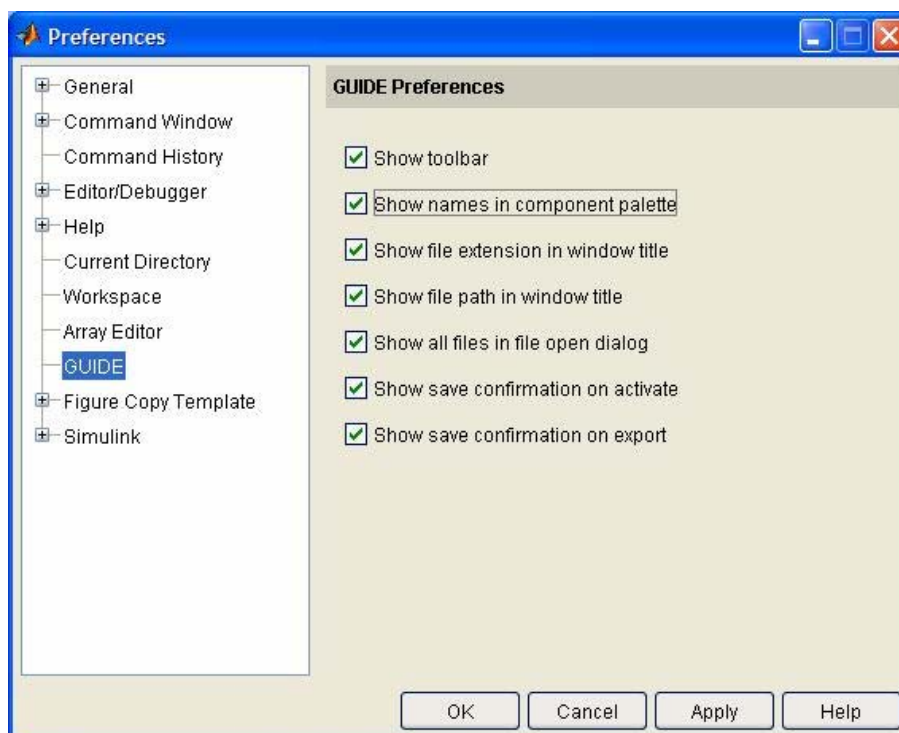
Obr. 2.9 Kontextové menu Layout Area GUI



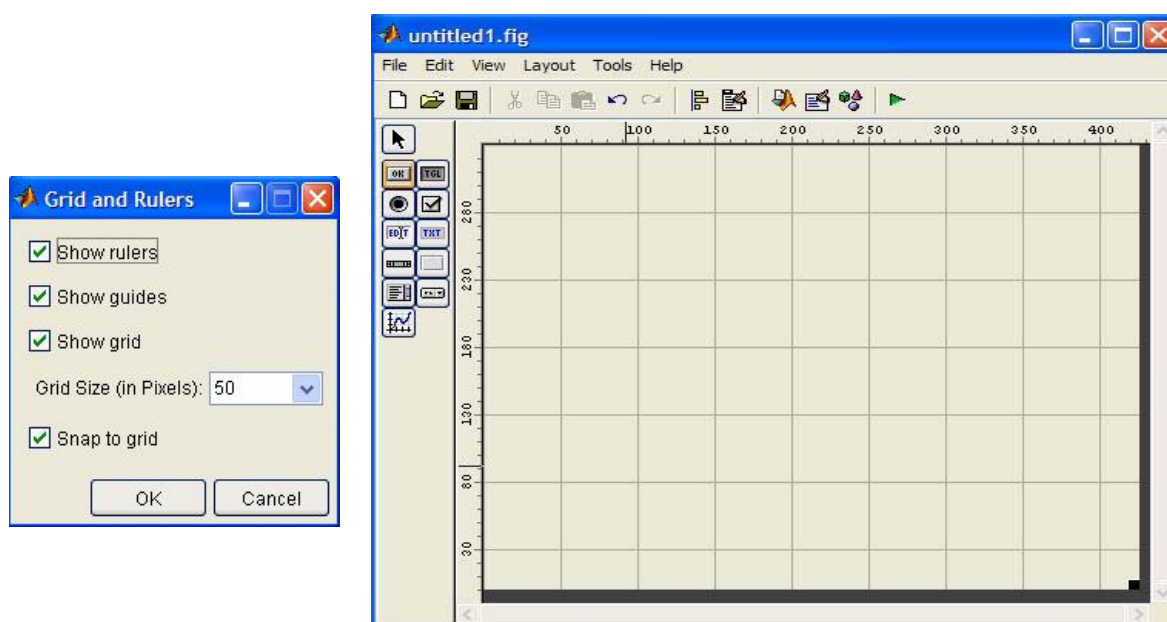
Obr. 2.10 Vyvolání preferencí v nastavení MATLABu

Druhou užitečnou položkou, kterou můžeme ovlivnit síť v *Layout Area*, je položka *Grid and Rules* v menu *Tools*. Po jejím zvolení se otevře následující okno, viz obr. 2.12.

Rules jsou pravítka. Stupnice je po padesáti pixelech. Toto nastavení lze v rozbalovacím menu změnit.



Obr. 2.11 Možnosti vlastního nastavení MATLABu



Obr. 2.12 Nastavení mřížky, vodících linek a měřítka

Guides jsou vodící linky (defaultně modré barvy). Na obrázku jsou vidět dvě vertikální. Získáme je vytažením z měřítka (*Ruler*). Najedeme-li myší na levý okraj *Layout Area*, změní se nám kurzor na opačnou dvojšipku. Podržíme-li nyní levé tlačítko a táhnete vpravo, vytáhnete si vodící linku. V místě, kde tlačítko pustíte, zůstane linka zafixovaná. Lze to opakovat víckrát a vytáhnout si vodících linek potřebné množství. Stejně tak vytažením z horního okraje *Layout Area* si můžeme vytáhnout požadovaný počet horizontálních linek. Pokud potřebujeme zafixovanou vodící linku přemístit, je to jednoduché. Najetím kurzoru myší na linku se nám tento změní na čtyřšipku, stisknutím levého tlačítka a tažením linku přemístíme a puštěním tlačítka se linka zafixuje na novém místě.

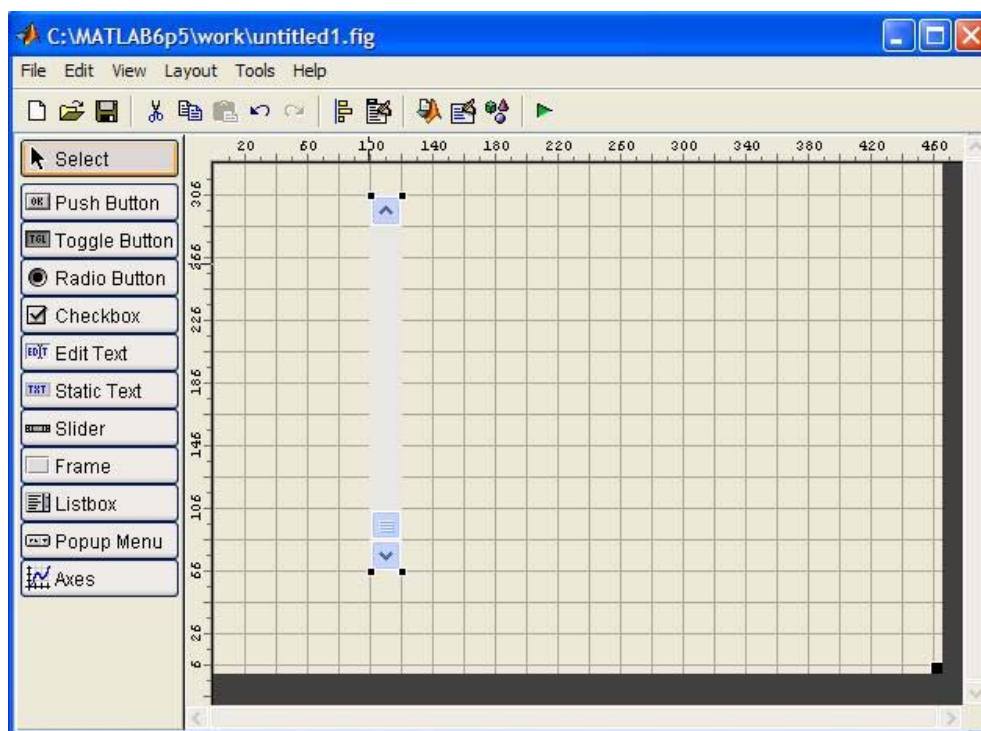
Ještě si všimněme dvou čárek v měřících. Na vodorovném měřítku při horním okraji *Layout Area*, na svislém měřítku při levém okraji *Layout Area*. To je pozice kurzoru v okamžiku sejmутí obrázku. Tyto čárky tedy ukazují okamžité souřadnice pohybujícího se kurzoru.

2.3 Tvorba grafických objektů pomocí GUIDE

Kliknutím levého tlačítka myší na příslušný objekt v *Component Palette* tento vybereme a pouhým přetažením jej umístíme na plochu *Layout Area*. Nebo na něj jen klikneme a následným kliknutím do *Layout Area* se nám tento na příslušném místě objeví. Pokud klikneme podruhé, objeví se nám stejný podruhé. Zbavíme se jej klávesou Delete. Abychom mohli pracovat s tímto objektem, posouvat jej po ploše, měnit rozměry, je třeba kliknutím na volbu *Select* nahoře v *Component Palette*, objekt „oživit“. Což je situace na obr. 2.13, kdy máme navíc zapnuto *Show Grids*, *Show Guides*, *Show Rulers* a *Grid Size* na *20 pixels*.

2.3.1 Slider (posuvník)

Posuvník se nám na plochu umístil jako svislý s implicitně nastavenými rozměry. Z obr. 2.13 je jasné, že uchopením za jeden ze čtyř vodících bodů lze rozměry libovolně měnit. Jakmile šířka přesáhne výšku, změní se na vodorovný.



Obr. 2.13 Posuvník umístěný na ploše

Kliknutím na ikonku *Activate Figure* (zelený trojúhelníček v liště ovládacích ikon zcela vpravo) se otevře dotaz, pokud jsme před tím nezaškrtnuli políčko *Don't tell me again* (je lepší v začátcích udržet procesy pod kontrolou). Potvrzením „Yes“ vzniknou soubory např. **untitled.fig**, kam se nám uloží vzhled a **untitled.m**, kam se nám uloží zdrojový kód (*GUI Setup Code*), popisující momentální situaci zobrazenou na *Layout Area*. Tento zdrojový kód se nám také okamžitě otevře v editoru zdrojových kódů.

Zapsáním názvu souboru – tedy **untitled** do *Command Window*, nebo spuštěním přímo z editoru (F5 nebo volba *Debug/Run*) se nám spustí takto vygenerované primitivní GUI.

Tento posuvník je již „živý“. Jezdcem lze myší jednoduše posouvat. Chceme-li změnit jeho vlastnosti, musíme si otevřít průvodce *GUIDE* – obr. 2.1, otevře se GUI s názvem **untitled.fig** a v něm si vyvolat (opět) – *File ... Open* soubor **untitled.fig** ze složky ...\\work.

2.3.2 Edit Text (přepisovatelné textové pole)

Spustíme opět průvodce GUIDE, viz obr. 2.1. Otevře se GUI s názvem **untitled.fig** a v něm si vyvoláme (opět) pomocí *File/Open* již vytvořený soubor, např. **slider2.fig**, z implicitní složky (např. . . \Work). Otevře se situace podobná obr. 2.13.

Klikneme nyní v *Component Palette* na položku *Edit Text* a přetáhneme ji myší do *Layout Area*. Uvolněním tlačítka myši na ploše zůstane políčko se čtyřmi vodícími čtverečky, kterými si ji zvětšíme do rozumné velikosti. Uložme si (*Save As ...*) situaci např. do **slider3.fig**.

Dokud je pole *Edit Text* „živé“, vyvolejme si *Property Editor*. V něm si najděte položku *String* a zapišme libovolný text.

2.3.3 Přehled jednotlivých Uicontrol objektů a jejich zpětnovazební kódy (Callback)

PUSH BUTTON

Jednostavové tlačítko. Akce se generuje jeho stlačením, čímž se změní navenek jeho vzhled. Uvolněním se vrací do původní nestlačené polohy.

Vlastnosti k nastavení jsou:

String – nápis na tlačítku,

Tag – jméno subfunkce v aplikačním M-souboru.

TOGGLE BUTTON

Dvoustavové tlačítko – přepínač. „Pamatuje si“ poslední polohu. Callback rutina musí uchovat aktuální polohu tlačítka. MATLAB nastavuje hodnotu (*Value*) rovnu buď **Max**, je-li přepínač stlačen (implicitně 1), nebo **Min**, je-li přepínač v nesepnuté poloze (implicitně 0).

```
function varargout = togglebutton1_Callback(h,eventdata,handles,varargin)
button_state = get(h, 'Value');
if button_state == get(h, 'Max') % stlačeno
elseif button_state == get(h, 'Min') % uvolněno
end
```

RADIO BUTTON

Podobá se funkcí zaškrťovacímu políčku (Check Box). Radiobutton má dva stav „vybrán – nevybrán“, což je uchováno ve vlastnosti *Value* „Max – Min“. Každému radiobuttonu musí být přidána příslušná subfunkce. Argumentem je matice obsahující handle ostatních radiobuttonů ve skupině, které musí být „nevybrán“.

```
function mutual_exclude(off)
set(off, 'Value',0)
```

Další volání je **mutual_exclude** generované z callback prvního ze čtyř radiobuttonů ve skupině (POZOR, na druhém řádku jsou hranaté závorky):

```
function varargout = radiobutton1_Callback(h,eventdata,handles,varargin)
off =[handles.radiobutton2, handles.radiobutton3, handles.radiobutton4];
mutual_exclude(off)
% pokračuje callback, implementující specifickou akci příslušného
radiobuttonu.
```

CHECK BOX

Zaškrťovací políčko generuje akci je-li vybráno a tento svůj stav uchovává ve vlastnosti *Value* „Max – Min“ (1 – 0). Určení momentálního stavu se děje dotazem na *Value*.

```
function varargout = checkbox1_Callback(h,eventdata,handles,varargin)
if (get(h, 'Value') == get(h, 'Max'))
% then „checkbox je vybrán“ proved' příslušnou akci
else
% „checkbox není vybrán“ proved' příslušnou akci
end
```

EDIT TEXT

Umožňuje uživateli vkládat nebo modifikovat textový řetězec. obsahuje text vložený uživatelem.

Vlastnost String obsahuje text vložený uživatelem.

```
function varargout = edittext1_Callback(heventdata,handles,varargin)
user_string = get(get(h, 'String'))
% pokračuje callback
```

STATIC TEXT

Užívá se nejčastěji jako název k **uicontrol** objektům, nebo ke sdělení informací.

SLIDER

Užívá numerický vstup v určitém rozsahu, čímž umožňuje uživateli pohybovat jezdcem. Pohyb se děje plynule tažením jezdce myši nebo skokově kliknutím do dráhy jezdce. Pozice jezdce je indikována numerickou hodnotou.

Příslušné vlastnosti jsou:

Value – Obsahuje numerickou hodnotu pozice posuvníku. Nebo může být tato pozice nastavena a ovládána z jiného místa programu. Pak musí zpětná vazba obsahovat příkaz:

```
slider_value = get(handles.slider1, 'Value');
```

Max – Definuje maximální hodnotu vlastnosti **Value**,

Min – Definuje minimální hodnotu vlastností **Value**,

SliderStep – Řídí jemnost změny hodnoty vlastnosti **Value**. Je specifikována vektorem se dvěma hodnotami. „Defaultní“ nastavení [0.01 0.10] znamená 1% krok při tažení jezdce myši a 10% krok při kliknutí myši do dráhy jezdce.

Aktuální absolutní hodnoty jsou závislé na rozsahu Min – Max.

FRAME

Rámy jsou obdélníkové oblasti zahrnující části okna *Figure*. Používají se k vizuálnímu ohraničení určitých skupin objektů **uicontrol** s výjimkou *Axes*. Nemají zpětnovazební volání. Jsou neprůhledné, objekty se umísťují na ně.

Přetáhneme-li objekty na *Frame*, jsou neviditelné (jsou umístěny pod *Frame*). Kliknutím pravého tlačítka na *Frame* a vybráním volby *Send to Back* se *Frame* umístí pod objekty a tyto jsou viditelné. *Bring to Front* má účinek opačný.

LIST BOX

Zobrazuje seznam, ze kterého je možno volit jednu nebo více položek. Vlastnost **String** obsahuje seznam řetězců, které se v nabídce zobrazují. První položka má index roven 1.

Vlastnost *Value* je vektor obsahující indexy řetězců vybraných položek. Implicitně je nastaveno zvýraznění první položky, je-li Listbox zobrazen poprvé. Je-li třeba, aby nebyla zvýrazněna žádná položka, pak nutno nastavit **Value** jako prázdnou, [].

Vlastnost **ListboxTop** definuje, která položka seznamu je nastavena jako horní, je-li položek tolik, že nemohou být zobrazeny všechny najednou. **ListboxTop** je index v matici řetězců, definovaný vlastností **String**. Jeho hodnota musí být mezi jedničkou a počtem řetězců. Není-li hodnota celočíselná, pak je rovna nejbližšímu nižšímu číslu.

Hodnoty vlastností **Min** a **Max** určují, zda je možno vybrat více položek, nebo jen jednu.

Je-li $\text{Max} - \text{Min} > 1$, pak je možný výběr více položek,

je-li $\text{Max} - \text{Min} < 1$, není výběr více položek možný.

Listbox rozlišuje mezi jednoduchým kliknutím a dvojklikem na položku a nastavuje podle toho vlastnost obrázku **SelectionType** na *normal* nebo *open*. MATLAB vyhodnocuje

zpětnou vazbu Listboxu po uvolnění tlačítka myši, nebo po uvolnění klávesy (včetně šipek), které mění vlastnost **Value**. To znamená, že příkazy zpětné vazby se vykonají po prvním kliknutí dvojkliku na jednotlivé položce. Je-li možný výběr více položek, je třeba vykonání odložit. V této situaci je třeba přidat další komponentu – *Push button* s funkcí *Hotovo (Done)* a naprogramovat její zpětnovazební rutinu k dotazu na **Value** Listboxu, namísto tvorby zpětné vazby pro Listbox.

Je-li aplikační M-soubor vygenerován automaticky průvodcem GUIDE, je třeba buď nastavit *Callback* jako prázdný řetězec (‘’) a odstranit *Callback* subfunkci z M-souboru, nebo nechat základ *Callback* subfunkce, s tím, že se vykoná hned po kliknutí na položku.

POPUP MENU

Po kliknutí na šipku zobrazuje seznam, ze kterého je možno volit jen jednu položku. Tím se trochu podobá sérii radiobuttonů.

Vlastnost **String** obsahuje seznam řetězců, které se v nabídce zobrazují. Vlastnost **Value** obsahuje index odpovídající řetězci vybrané položky. První položka v seznamu nabídek má index 1. Není-li popup menu otevřeno, je zobrazena položka dle indexu obsaženého ve **Value**.

Zpětná vazba kontroluje index vybrané položky a přepínacím (*switch*) příkazem provede akci založenou na **Value**. Je-li obsah popup menu fixní lze použít tento přístup:

```
function varargout = popupmenu1_Callback(h,eventdata,handles,varargin)
val = get(h, 'Value');
switch val
case1
% je vybrána první položka
case2
% je vybrána druhá položka
% atd.
```

Zpětná vazba obdrží aktuální řetězec vybraný v popup menu. Použije index z **Value** k vyhledání položky v seznamu řetězců. Tento přístup může být užitečný, když program dynamicky nahrává obsah popup menu založený na uživatelské akci. Je nutná konverze hodnoty indexu z **Value** do řetězce.

```
function varargout = popupmenu1_Callback(h,eventdata,handles,varargin)
val = get(h, 'Value')
string_list = get(h, 'String');
selected_string = string_list(val); % konverze na řetězec
% atd.
```

AXES

Umožňují zobrazit a editovat grafy.

FIGURE

Je okno obsahující GUI ať už naprogramované, nebo vytvořené průvodcem.

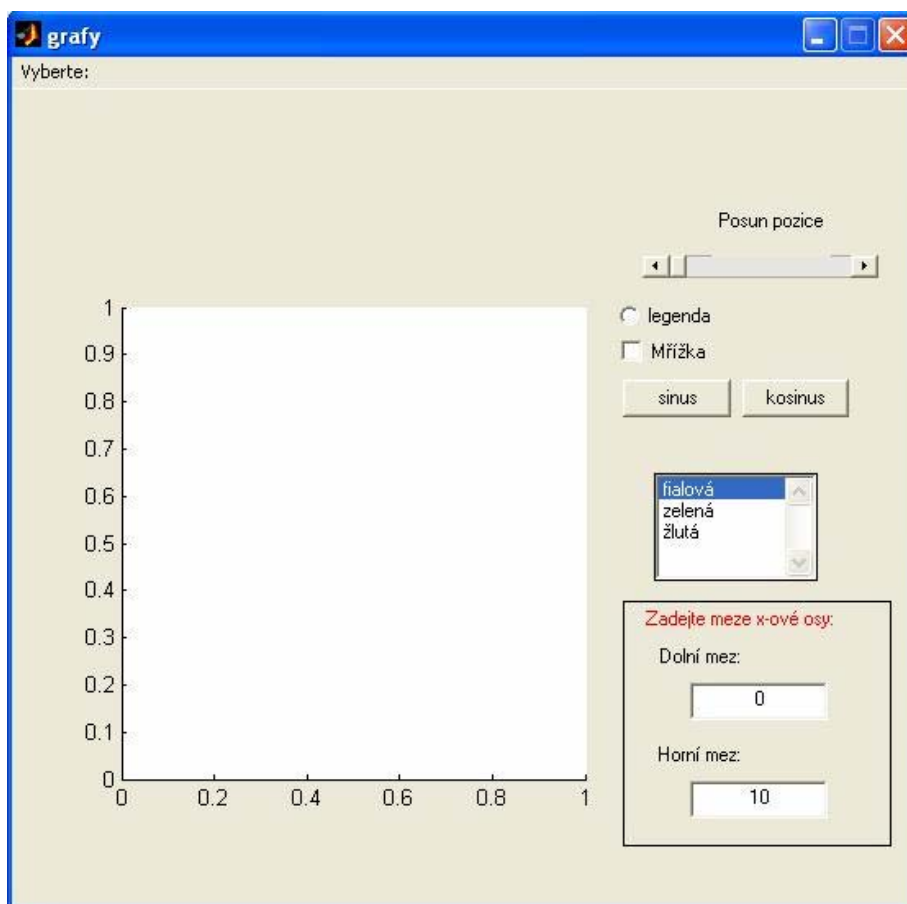
2.3.4 Zarovnávání objektů

Možnost zarovnání objektů plní společně s rámy (*Frame*) dvě důležité funkce. Jednak zpřehledňují celou aplikaci a na druhé straně plní funkci estetickou. Nic nám nepomůže propracovaná a bezvadně fungující aplikace, jsou-li příbuzné objekty každý jinak velký a po okně aplikace jsou poházené.

Stejné rozměry stejných objektů zajistíme buď nastavením stejných hodnot v každém *Property Inspectoru* nebo jednodušeji kliknutím pravým tlačítkem na objekt a volbou *Duplicate*. Zarovnáváme vždy více objektů, musí tedy být všechny „oživeny“, musí být všechny vybrány. Klikneme do *Layout Area* a tažením se vytáhne rámeček okolo objektů. Puštěním tlačítka se všechny naráz vyberou a můžeme je zarovnat.

3 UKÁZKOVÝ PROGRAM POMOCÍ GUIDE MATLABU

Jako ukázkový program vytvořený pomocí GUIDE v MATLABu jsem si vybral jednoduchou aplikaci pro vykreslení grafů funkcí sinus a kosinus (viz obr 3.2).



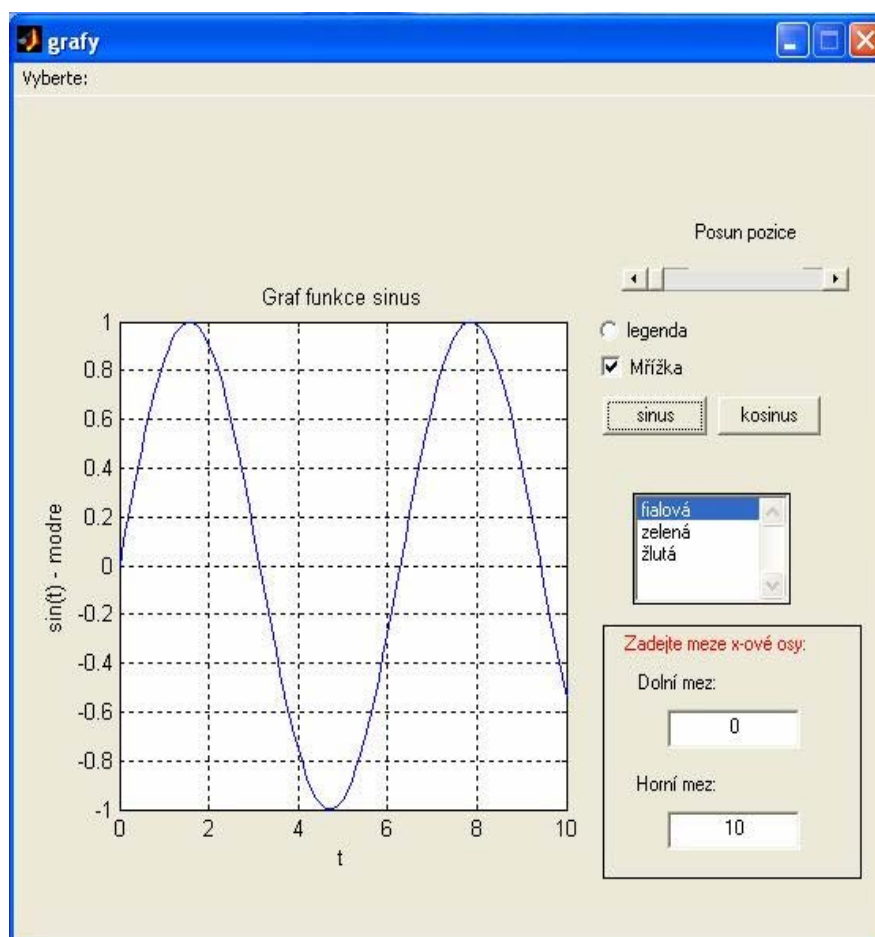
Obr. 3.1 Okno programu

Jak je zřejmé z obr. 3.1, jedná se o okno, které obsahuje pole pro vykreslení grafu funkce. Dalšími prvky jsou:

- tlačítko (*Push Button*) pro zvolení dané funkce,
- posuvník (*Slider*) pro změnu pozice pole grafu,
- políčko (*Radio Button*) pro zvolení legendy ke grafu,
- zaškrtnutí políčko (*Check Box*) pro zapnutí nebo vypnutí mřížky v poli grafu,
- seznam (*List Box*), ze kterého je možné vybrat barvu křivky,

- políčka (*Edit Text*), do kterých můžeme zadávat hodnoty horní a dolní meze x-ové osy.

Funkce všech těchto prvků jsou popsány v předchozí kapitole 2.3 Tvorba grafických objektů pomocí GUIDE.



Obr. 3.2 Program Grafy s vykreslenou funkcí sinus

4 NÁSTROJ PROFILER

Profiler je nástroj pro zjištění času stráveného spouštěním funkcí M-souboru. Pomocí tohoto nástroje můžeme zjistit, kde se dá optimalizovat kód námi vytvořeného programu, což vede ke zvýšení výkonu. Profiler vytvoří zprávu *Profile Summary*, která nám poskytne dobu a četnost vyvolávání funkcí, které běžely v době, kdy byl spuštěn nástroj Profiler. Z tohoto shrnutí lze poznat funkci, jejíž rychlost chcete zlepšit, protože čas nebo počet volání na funkci je vysoký. Kliknutím na jméno funkce se nám objeví podrobná zpráva pro danou funkci. Zprávu můžeme také vytisknout kliknutím na ikonu tisku [3, 5, 8].



Obr. 4.1 Okno nástroje Profiler

ZÁVĚR

Má bakalářská práce se skládá ze 4 kapitol pojednávajících o programu MATLAB, jeho grafickém prostředí GUIDE, ukázkovém programu pomocí GUIDE a nástroji Profiler. Tyto kapitoly tvoří navazující soubor informací.

Cílem této bakalářské práce je tedy obeznámit se základními funkcemi programu MATLAB®. Je zde také podrobně popsána funkce grafického prostředí GUIDE. Důležitou částí mé bakalářské práce je tvorba ukázkového programu v grafickém prostředí GUIDE. V poslední části jsem se zaměřil na nástroj Profiler a jeho funkci.

SEZNAM POUŽITÉ LITERATURY

Skripta a knihy

- [1] ZAPLATÍLEK, K., DOŇAR, B.: *MATLAB pro začátečníky*. 1.vyd. Praha: BEN-technická literatura, 2003. 144 s. ISBN 80-7300-095-4.
- [2] ZAPLATÍLEK, K., DOŇAR, B.: *MATLAB - tvorba uživatelských aplikací*. 1. vyd. Praha: BEN-technická literatura, 2004. 216 s. ISBN 80-7300-133-0.
- [3] MATLAB: *Getting Started With MATLAB, Version 7*, MathWorks Inc., Natick, USA, 2004.
- [4] MATLAB: *Creating Graphical User Interfaces, Version 7*, MathWorks Inc., Natick, USA, 2004.
- [5] MATLAB: *Function Reference, Version 7*, MathWorks Inc., Natick, USA, 2004.
- [6] DUŠEK, F.: *Matlab a Simulink úvod do používání*. Bratislava: Univerzita Pardubice, 2000. ISBN 80-7194-273-1.

Články a materiály dostupné na internetu

- [7] Humusoft – Web Pages [online] [cit. 14. dubna 2005] Dostupné na internetu: <<http://www.humusoft.cz>>.
- [8] Mathworks – Web Pages [online] [cit. 21. března 2005] Dostupné na internetu: <<http://www.mathworks.com>>.
- [9] Matlab – Web Pages [online] [cit. 15. února 2005] Dostupné na internetu: <<http://k315.feld.cvut.cz>>.
- [10] Matlab – Web Pages [online] [cit. 24. září 2004] Dostupné na internetu: <<http://vyukavt2.vscht.cz>>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

HTML	zkratkou názvu <i>HyperText Markup Language</i> – jazyk, kterým jsou psány zdrojové kódy webových stránek. Ve zdrojovém textu, který si může každý na svém prohlížeči zobrazit, najdeme celý kód stránky spolu s odkazy na soubory s obrázky a dalšími objekty na stránce. Hypertext – je text obsahující odkazy na další dokumenty nebo stránky, které mohou být volbou odkazu zobrazeny.
MS-Word	textový editor Microsoft Word
PPAŘ	Programová podpora automatického řízení
trans.	transpozice
uicontrol	vytvoření tlačítek přepínačů a oken dialogů

PŘÍLOHY

P 1 : Zdrojový kód programu Grafy.m pomocí GUIDE v MATLABu

```
function varargout = grafy(varargin)
% GRAFY M-file for grafy.fig
%   GRAFY, by itself, creates a new GRAFY or raises the existing
%   singleton*.
%
%   H = GRAFY returns the handle to a new GRAFY or the handle to
%   the existing singleton*.
%
%   GRAFY('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in GRAFY.M with the given input arguments.
%
%   GRAFY('Property','Value',...) creates a new GRAFY or raises the
%   existing singleton*. Starting from the left, property value pairs
are
%   applied to the GUI before grafy_OpeningFunction gets called. An
%   unrecognized property name or invalid value makes property
application
%   stop. All inputs are passed to grafy_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help grafy

% Last Modified by GUIDE v2.5 22-Feb-2005 14:20:42

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @grafy_OpeningFcn, ...
                  'gui_OutputFcn',  @grafy_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before grafy is made visible.
function grafy_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to grafy (see VARARGIN)

% Choose default command line output for grafy
handles.output = hObject;
handles.dolnimez=0;
```

```
handles.hornimez=10;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes grafy wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = grafy_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in tlacitko_sin.
function tlacitko_sin_Callback(hObject, eventdata, handles)
% hObject handle to tlacitko_sin (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
kontrola(hObject, eventdata, handles)
t=handles.dolnimez:.1:handles.hornimez;
sinx=sin(t);
hold off
plot(t,sinx,'b-')
title('Graf funkce sinus')
xlabel('t')
ylabel('sin(t) - modre')
grid on
handles.t=t;
handles.sinx=sinx;
handles.stav=1;
guidata(hObject, handles);
hold on

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
kontrola(hObject, eventdata, handles)
t=handles.dolnimez:.1:handles.hornimez;
cosx=cos(t);
hold off
plot(t,cosx,'ro-')
title('Graf funkce kosinus')
xlabel('t')
ylabel('cos(t) - cervene')
grid on
handles.t=t;
handles.cosx=cosx;
handles.stav=2;
guidata(hObject, handles);
hold on

% -----
function Untitled_1_Callback(hObject, eventdata, handles)
% hObject handle to Untitled_1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% -----
function Untitled_4_Callback(hObject, eventdata, handles)
```

```

% hObject      handle to Untitled_4 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% -----
function Untitled_5_Callback(hObject, eventdata, handles)
% hObject      handle to Untitled_5 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
grid on

% -----
function Untitled_6_Callback(hObject, eventdata, handles)
% hObject      handle to Untitled_6 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
grid off

% -----
function mrizka_Callback(hObject, eventdata, handles)
% hObject      handle to mrizka (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject      handle to listbox1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject      handle to listbox1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listbox1 contents as
cell array
%       contents{get(hObject,'Value')} returns selected item from
listbox1

barva=get(hObject,'Value');
if handles.stav==1
    switch barva
        case 1
            plot(handles.t,handles.sinx,'m')
        case 2
            plot(handles.t,handles.sinx,'g')
        case 3
            plot(handles.t,handles.sinx,'y')
        otherwise
            plot(handles.t,handles.sinx,'k')
    end
elseif handles.stav==2
    switch barva

```

```

        case 1
            plot(handles.t,handles.cosx,'mo-')
        case 2
            plot(handles.t,handles.cosx,'go-')
        case 3
            plot(handles.t,handles.cosx,'yo-')
        otherwise
            plot(handles.t,handles.cosx,'k')
    end
end

% -----
function Untitled_7_Callback(hObject, eventdata, handles)
% hObject      handle to Untitled_7 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
!explorer http://www.utb.cz

% --- Executes during object creation, after setting all properties.
function dolni_CreateFcn(hObject, eventdata, handles)
% hObject      handle to dolni (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function dolni_Callback(hObject, eventdata, handles)
% hObject      handle to dolni (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of dolni as text
%         str2double(get(hObject,'String')) returns contents of dolni as a
double
handles.dolnimez=str2double(get(hObject,'String'));
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit2_Callback(hObject, eventdata, handles)
% hObject      handle to edit2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

```

```
% Hints: get(hObject,'String') returns contents of edit2 as text
%         str2double(get(hObject,'String')) returns contents of edit2 as a
double
handles.hornimez=str2double(get(hObject,'String'));
guidata(hObject, handles);

% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject     handle to checkbox1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox1

hodnota=get(hObject,'Value')
if hodnota==0
    grid off
elseif hodnota==1
    grid on
end

% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject     handle to radiobutton1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton1

hodnota=get(hObject,'Value')
if hodnota==0
    legend('nezatrzeno')
elseif hodnota==1
    legend('zatrzeno')
end

function kontrola(hObject, eventdata, handles)
% prevede meze
dolni_Callback(hObject, eventdata, handles);
edit2_Callback(hObject, eventdata, handles);
if (handles.hornimez<handles.dolnimez)
    okno=questdlg('Máte horní mez nižší než dolní mez! Chcete čísla
automaticky přehodit?',...
        'Otázka','Ano','Ne','Ano');
    switch okno
        case 'Ano'
            pomocna=handles.hornimez;
            handles.hornimez=handles.dolnimez;
            handles.dolnimez=pomocna;
            guidata(hObject,handles);
            set(handles.dolni,'String',num2str(handles.dolnimez));
            set(handles.edit2,'String',num2str(handles.hornimez));
        case 'Ne'
            warndlg('Vybral(a) jste ne, napište proto meze znovu, jinak
se zavolá předchozí dialog.','...
            'Hláška')
    end %switch
end %if

% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject     handle to slider1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
```

```
% handles      empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject      handle to slider1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
slider

% min_sld=get(hObject,'Min') %==0
% max_sld=get(hObject,'Max') %==1
pozice=8+10*get(hObject,'Value')
set(handles.axes1,'Position',[13.4 pozice 56.6 22.153846153846153])
set(handles.text4,'String',num2str(pozice))
```

SEZNAM OBRÁZKŮ

Obr. 1.1 Knihovna toolboxů Simulink.....	8
Obr. 1.2 Zobrazení sekvence grafů.....	20
Obr. 2.1 Tři způsoby vyvolání GUIDE.....	23
Obr. 2.2 Základní vzhled Průvodce tvorbou GUI (GUIDE) o jeho spuštění	24
Obr. 2.3 Tlačítka (ikony) v horní části otevřeného okna GUIDE	24
Obr. 2.4 Možnosti použití nástroje Alignment Tool.....	25
Obr. 2.5 Spuštění Menu Editoru	26
Obr. 2.6 Okno tzv. Property Inspectoru – okno pro řízení vlastností GUI.....	26
Obr. 2.7 Spuštění prohlížeče objektů (Object Browser).....	27
Obr. 2.8 Uložení vzhledu GUI.....	27
Obr. 2.9 Kontextové menu Layout Area GUI	28
Obr. 2.10 Vyvolání preferencí v nastavení MATLABu	28
Obr. 2.11 Možnosti vlastního nastavení MATLABu	29
Obr. 2.12 Nastavení mřížky, vodících linek a měřítka	29
Obr. 2.13 Posuvník umístěný na ploše	31
Obr. 3.1 Okno programu.....	38
Obr. 3.2 Program Grafy s vykreslenou funkcí sinus.....	39
Obr. 4.1 Okno nástroje Profiler	40

SEZNAM TABULEK

Tabulka I. Základní matematické operace	13
Tabulka II. Speciální proměnné	13
Tabulka III. Komplexní čísla	14
Tabulka IV. Exponenciální funkce	14
Tabulka V. Transformace souřadnic	14
Tabulka VI. Ostatní funkce	15
Tabulka VII. Zobrazení dat	16
Tabulka VIII. Příkazy používané pro psaní skriptů	17
Tabulka IX. Příkazy pro manipulaci se soubory	17
Tabulka X. Způsob zápisu vektorů	18
Tabulka XI. Seznam funkcí pro práci s maticemi	19

SEZNAM PŘÍLOH

P 1 : Zdrojový kód programu Grafy.m pomocí GUIDE v MATLABu	44
--	----