We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

6,900
Open access books available

186,000
International authors and editors

200M
Downloads

154
Countries delivered to

Our authors are among the

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Tips and tricks for programming in Matlab

Karel Perutka
Tomas Bata University in Zlin, Faculty of Applied Informatics
Czech Republic, European Union

1. Introduction

Matlab is the software developed by the MathWorks, Inc., Natick, USA. In 1984, the first version appeared. Software was primarily used only for the mathematical computation enabling the computation of complicated matrix equations and their systems. All major functions can directly use the matrix as the input. From that year, the software is still under development enlarging the area of the users every year. Matlab became the standard in the area of simulation and modelling and it is used by the researchers and students at universities mainly in the areas of Control Engineering, Power Plant Systems, Aerospace, Bioinformatics, Economics and Statistics. In comparison to other software such as Mathematica or Maple, Matlab has several advantages. Let us mention some. Its open architecture enables sharing all source code among the user community and several different areas are solved and the solution appears usually as a new toolbox. Simulink is the important Matlab enlargement which simplifies the computation very much. You just drag and drop the blocks to the new window from the block libraries and connect them and run the model. Matlab is used not only at universities but also in practice, for instance by NASA or General Motors. Most Matlab users are from the major world countries, such as USA, Japan, China, and India. Nice book was written by Hanselman and Littlefield (Hanselman and Littlefield, 2005). And interesting paper about teaching using Matlab was written by Perutka and Heczko (Perutka & Heczko, 2007). This chapter provides some chosen tips and tricks to the beginners in Matlab programming and should ease the first steps in programming. These tips and tricks are based on the experience of chapter author with teaching Matlab programming in the undergraduate curriculum for several years. The author mostly worked in MATLAB version 6.5, for which tips are. They are unsorted and each chapter provides one tip or trick only.

2. Placing picture as dialog background

Being bored from standard look of GUI created in Matlab? If you create dialog in Matlab using GUIDE or set of functions `figure`, `uimenu`, `uicontrol` and `axes`, the dialog background is usually based on the settings of the system, for example in older Microsoft Windows it was grey – Windows Classic Theme. However, if you need to have your picture as the figure background, there is possible to use the following solution. Such example is

shown in figure 1. The dialog in this figure contains only 3 grey pushbuttons (`online ident.`, `Gr`, `>>`) and background picture.

What to do first? You have to draw the picture in the extern software, for example in Corel DRAW! and save it in one of the formats that Matlab supports, for instance as JPG or BMP. You can get the list of supported formats from the Matlab Help Dialog. Don't forget to write down the width and height of the picture or their ratio. Create new dialog using the command `figure` or by GUI. Set the size of the new window in accordance with the picture width and height or their ratio. How? If you working with GUIDE, double-click the window in GUIDE and Property Inspector should appear. Change the *Units* property on *Pixels* and after that change in the *Position* property the third and fourth value to the width and height of the figure or keep these values in the ratio of the picture you would like to show. If you created the dialog using `figure` function in M-file, include in the list of properties *Units* and *Position* and set them in similar way as was described for the GUI created by GUIDE. Now you need to load the figure in the Matlab Workspace, create axes and put the figure inside them. This tip is based on work of Perutka (Perutka, 2007). Inspire yourself by the following code which will be commented

```
1 STCssu=imread('STCssu.jpg','jpg');
2 axes('Position',[0 0 1 1]);
3 image(STCssu);
4 axis off;
5 clear STCssu
```
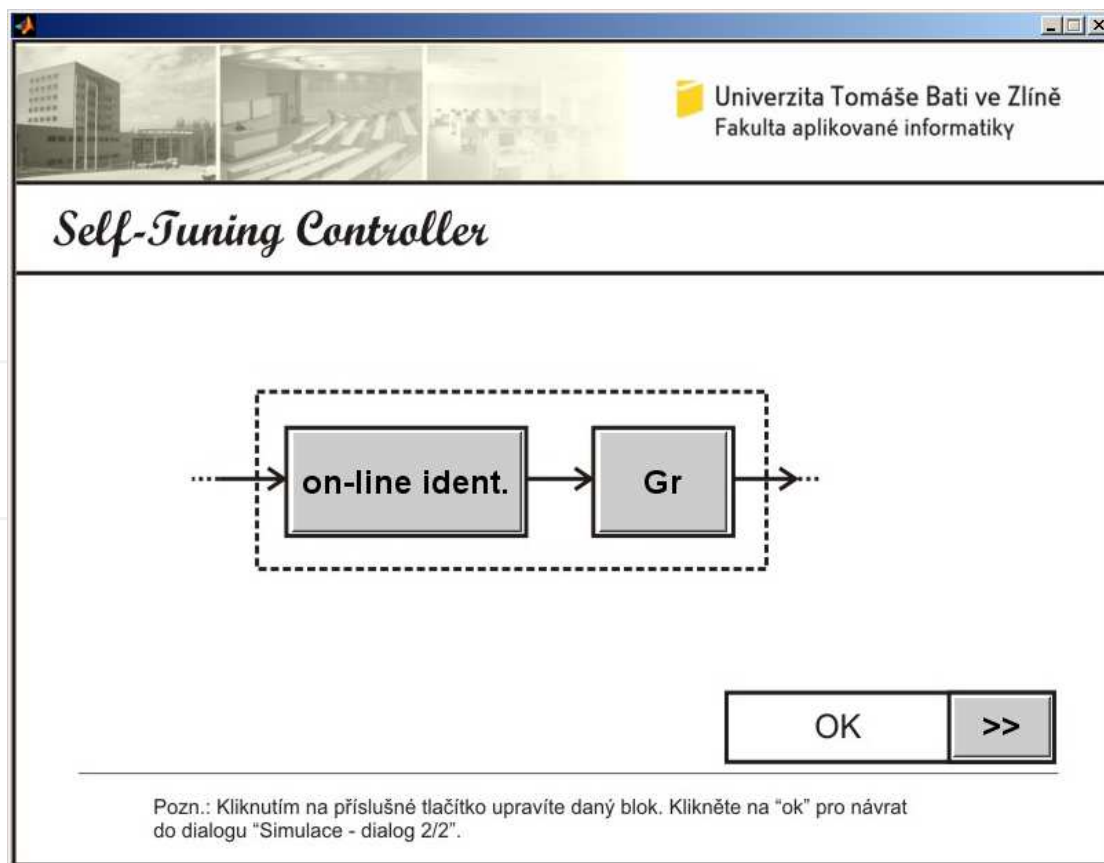

Fig. 1. Dialog with picture as its background

This code should be placed in the file where the figure is defined. The line 1 of the code is responsible of loading the picture in the Matlab Workspace using the function image. In this case, *STCssu.jpg* is the name of picture file and the picture is loaded to the *STCssu* variable. Line 2 creates the axes with their range just for all dialog area. Command in line 3 draws the image from the *STCssu* variable to the axes created by line 2. The axes description is hidden by the command on line 4. If you don't re-draw the dialog, you can delete the variable from the Matlab Workspace, as it is shown in line 5. But if you re-call the dialog, don't use line 5. For the dialog re-calling, lines 2 to 4 are enough.

3. Short online help for every object in dialog

Imagine the situation depicted in figure 2. There is a short text "If you click this button, you will open the dialog of setting the controller parameters." in the box. This text is shown if you keep the cursor on the button *Gr* for a while. This might be useful especially in the case of two sorts of objects in dialog (button or edit text) to provide necessary short information what should be written as the text (edit text) or what will happen when the button is pressed. But short information can be displayed over every object in dialog which has the property *TooltipString*.
If you created your dialog by GUIDE, open it in GUIDE again and double click the object for which you would like to create the short help.
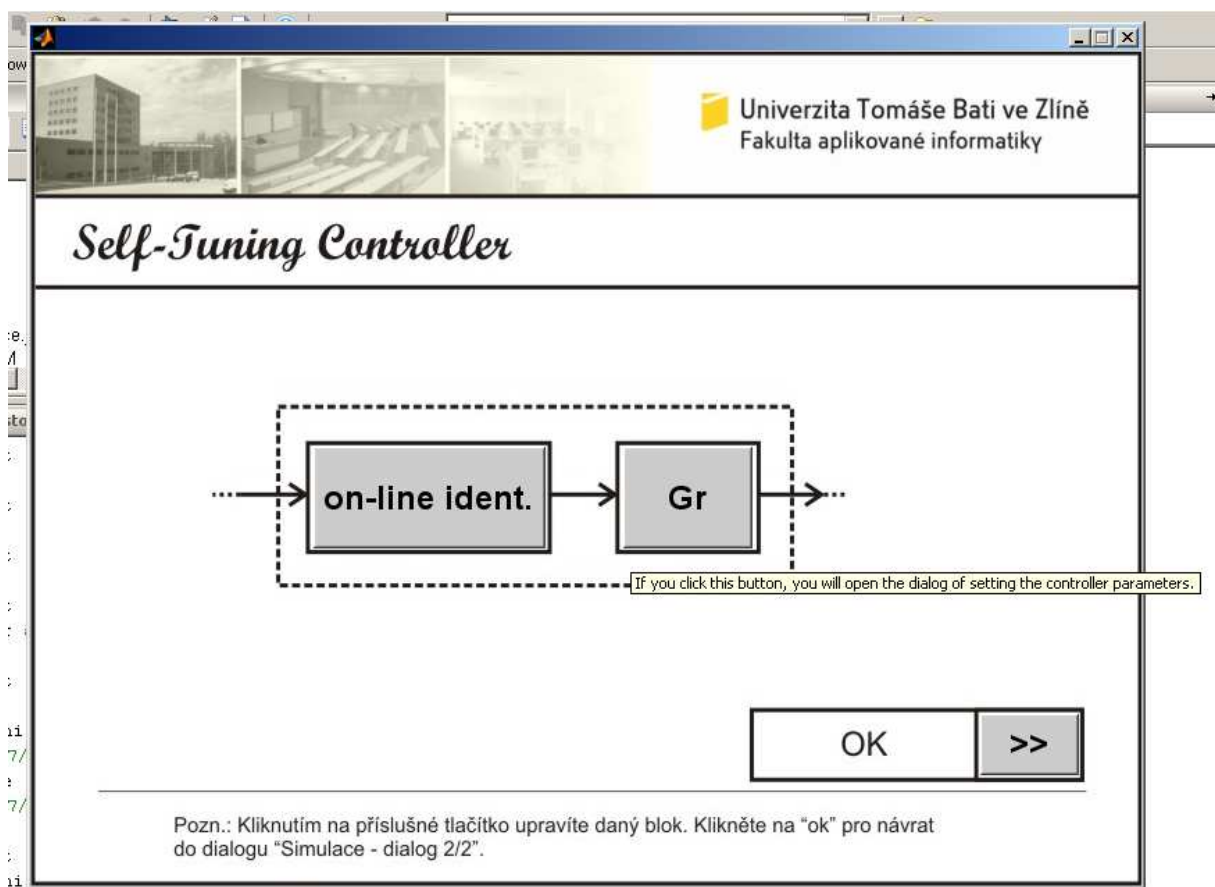

Fig. 2. Dialog with short help

The Property Inspector dialog should appear, find the property denoted as *TooltipString* and write some string as the input. The string you write will appear as the short help for the object after calling the dialog again.

If you created the object by the *uicontrol* function, just read the sample code below

```
6 hButtonClose = uicontrol('Style', 'pushbutton',...
7                 'String', 'Close',...
8                 'Parent', hFigure,...
9                 'Tag', 'tButtonClose',...
10                'Units', 'pixels',...
11                'Position', [560 75 100 30],...
12                'TooltipString', 'Press the button for exit.',...
13                'Callback', 'closeIt');
```

The code in lines 6 to 13 presents the definition of pushbutton object in the dialog created by commands in M-file. The *TooltipString* property of *uicontrol* is defined in line 12, the text "Press the button for exit." is shown as a short help when the cursor will be on the button "Close".

4. More pictures in one dialog and their control

It is sometime necessary to place more than one picture in the dialog, have a look at figure 3. The dialog in figure 3 includes 9 JPG pictures. The digital clock consists of 8 pictures, 6 pictures are the digits of the clock. The background of analogue clock is the remaining picture. These pictures are shown similarly as is described in "Placing picture as dialog background", it means the pictures are drawn using image function to pre-defined axes. The dialog in figure 3 was created as a set of commands in M-file and it is defined by lines 14 to 24, the *Render* and *DoubleBuffer* properties should be set according to lines 23 and 24.

```
14 hFigure = figure('Units', 'pixels',...
15                 'Position', [100 50 770 690],...
16                 'MenuBar','None',...
17                 'Name', 'Hodiny',...
18                 'Tag', 'tWindow',...
19                 'NumberTitle', 'off',...
20                 'Resize', 'off',...
21                 'Visible', 'off',...
22                 'BackingStore', 'off',...
23                 'Renderer', 'painters',...
24                 'DoubleBuffer', 'on');
```

When figure is defined, all axes objects are created, example for one axes object is shown in lines 25 to 33

```
25 hAnalog = axes('Units', 'pixels',...
26         'Position', [30 210 460 460],...
27         'Visible', 'on',...
28         'Parent', hFigure,...
29      'Tag', 'tAnalog',...
30      'XTickLabelMode', 'manual', 'YTickLabelMode', 'manual',...
```

```
31      'XColor', 'k', 'YColor', 'k',...
32      'DrawMode', 'fast',...
33      'Color', 'k');
```

This step is followed by step in which all images are read into Matlab Workspace using the image function, the image file are in the same directory as the source code, for one image the code will be

```
34      File_Analog = imread('clock.jpg');
```
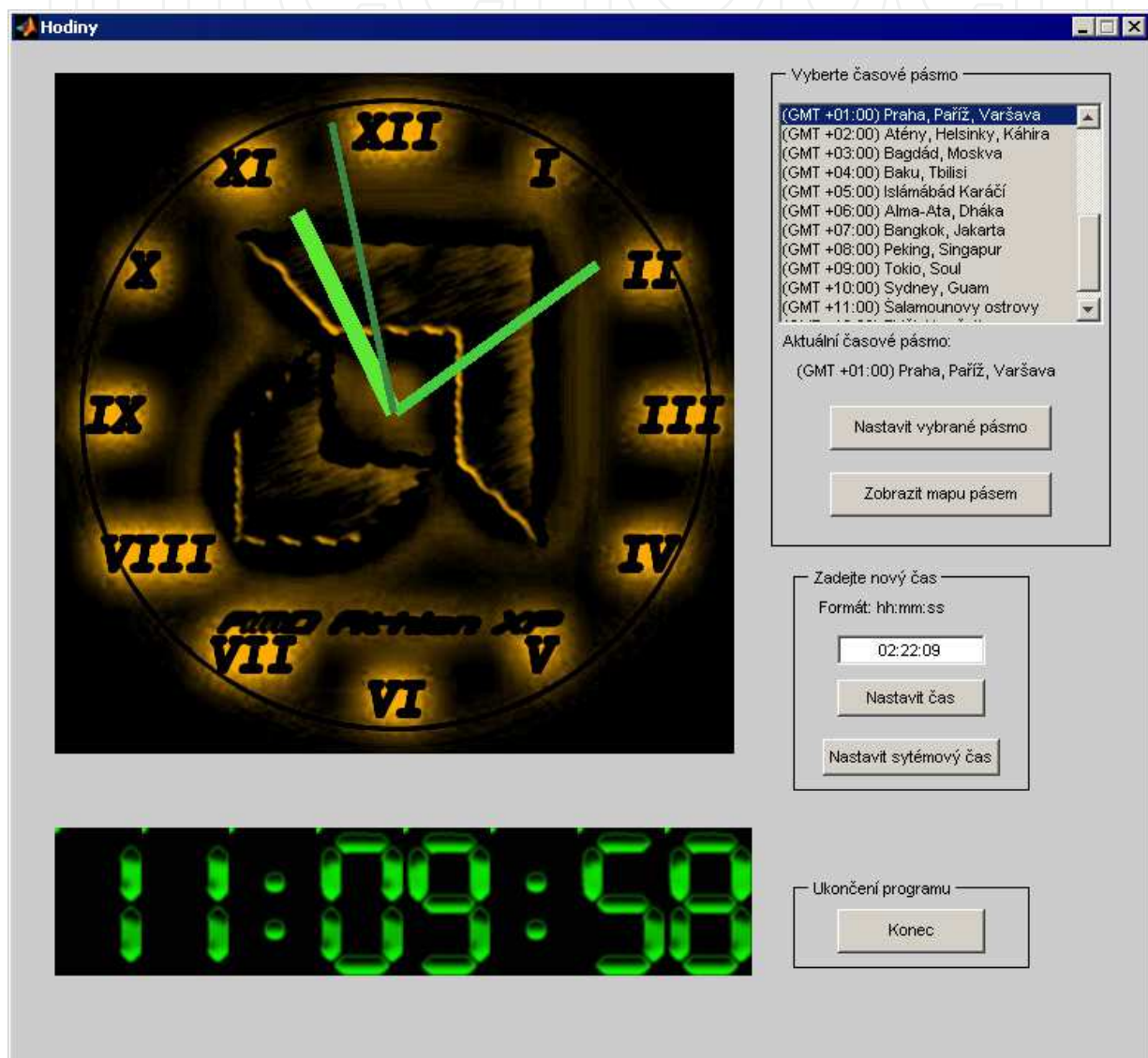

Fig. 3. Dialog with more pictures

Now have a careful look at the following source code, lines 35 – 39

```
35      set(0, 'CurrentFigure', hFigure);
36      set(hFigure, 'CurrentAxes', hAnalog);
37      image(File_Analog);
```

```
38      axis image;
39      axis off;
```

The source code on lines 35 – 39 shows how to switch among several axes in one dialog. Line 35 gives us an example where the dialog identified by hFigure, line 14, is set as the active one. And line 36 shows us the example of setting one of the axes as the active one. The axes are in the dialog, which is identified by hFigure, line 14, and the axes are identified by hAnalog, line 25. Now you show the chosen picture, line 37, which was read by line 34. Finally, you place the axis just to the borders of the image, line 38, and you hide the ticks, line 39. This tip is based on the example provided by Perutka (Perutka, 2005).

5. Button with your picture

Figure 4 shows the green dialog with two options to be chosen, they are both yellow and black. If you click "Stromek" or "Kytka", new dialog appears. The dialog in figure 4 consists of 3 axes objects to which the pictures are drawn, one axes object shows the background and 2 axes objects show pictures as "buttons".
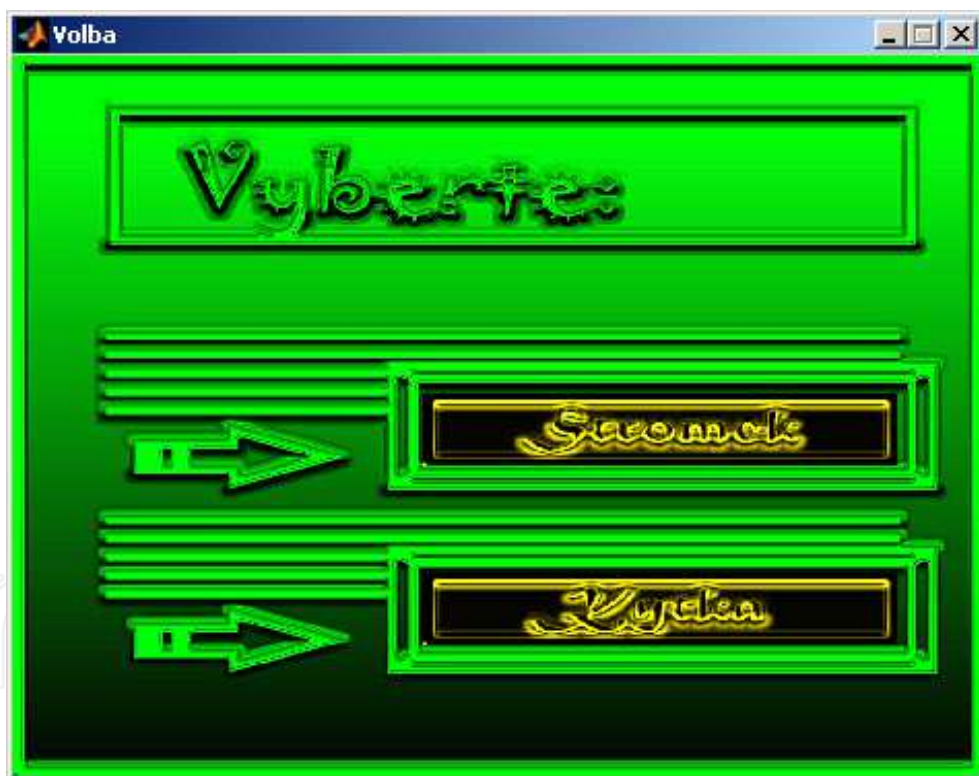

Fig. 4. Dialog with "2 buttons" with our picture.

Now let us look at the source code, lines 40 – 64. These lines provide the full source code for figure 4. See line 59 or 63. There is another syntax of image function in comparison to line 3. The ButtonDownFcn property is used, its value is set on the name of function or file, which should be called if the picture or axes are clicked. For example if you click on "Kytka", there will be called the file *kytka.m* according to line 63.

```
40 hFigure = figure('Units','pixels',...
41             'Position',[160 160 470 350],...
42             'MenuBar','None',...
43             'Name','Volba',...
44             'Tag','tOkno',...
45             'NumberTitle','off',...
46             'Resize','off',...
47             'Visible','on',...
48             'BackingStore','off',...
49             'Renderer','painters',...
50             'DoubleBuffer','on');
51     hlavniObr=imread('vyberte.jpg');
52     prvniObr=imread('stromek.jpg');
53     druhyObr=imread('kytka.jpg');
54     hlavniAxes=axes('Position',[0 0 1 1]);
55     image(hlavniObr);
56     axis off
57     prvniAxes=axes('Units','pixels',...
58         'Position',[200 152 230 35]);
59     image(prvniObr,'ButtonDownFcn','stromek');
60     axis off
61     druhyAxes=axes('Units','pixels',...
62         'Position',[200 65 230 35]);
63     image(druhyObr,'ButtonDownFcn','kytka');
64     axis off
```

6. New picture on button click

This task is connected with the previous example, see similar figure 5 and lines 65 to 71. If you click "the button", the button changes the color for a while, i.e. for 0.5 s, line 69. Actually, you set the axes, which will be "the button", as active, line 65. You read the image to the Matlab Workspace variable, line 66, and draw it in the selected axes, line 67, and hide the description of axes, line 68. To keep the illusion of the button, you draw the previous picture back, line 70, together with the option to be clicked again, i.e. there is ButtonDownFcn property included, line 70, there will be called the file *kytka.m* if you click "the button".

```
65     set(hFigure,'CurrentAxes',druhyAxes);
66     cervKytka=imread('kytka1.jpg');
67     image(cervKytka);
68     axis off
69     pause(0.5);
70     image(druhyObr,'ButtonDownFcn','kytka');
71     axis off
```
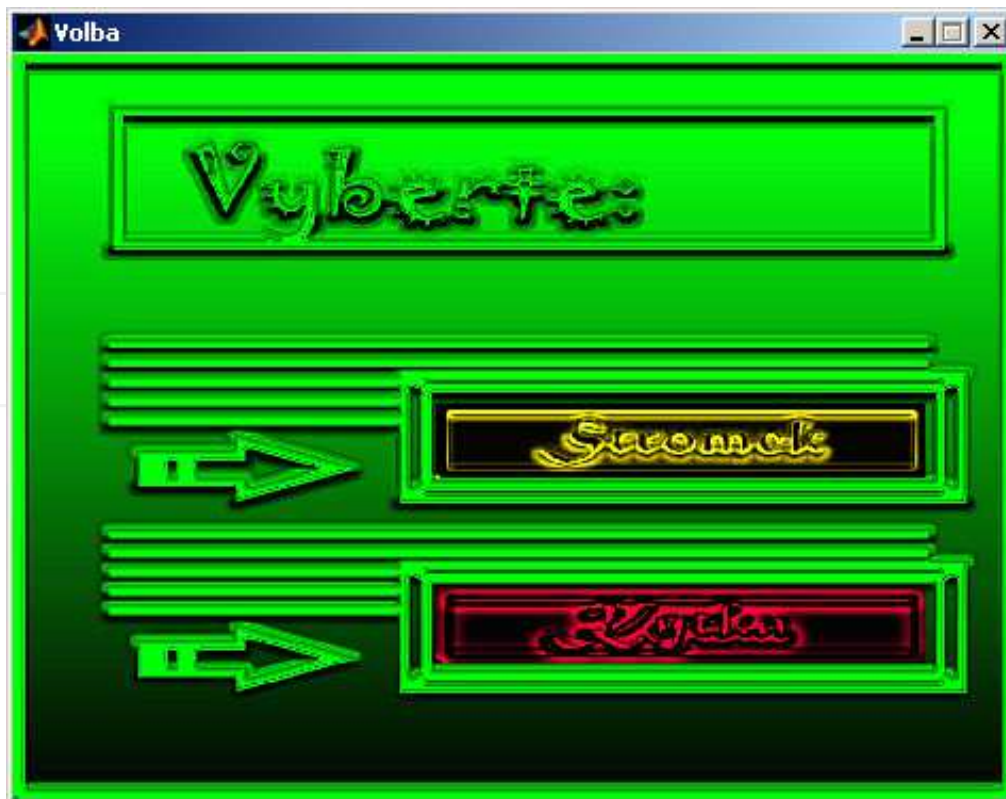
Fig. 5. New picture on "button" click.

7. Set of buttons with your pictures in dialog, each button is pressed on given key

Typical task for simple menu controlled by keyboard is the following one. You have a list of menu items such as in figure 6. The list is controlled by 3 keys. First key is used for moving up. One press means one item up. Second key is used for moving down and the last key selects the menu item. Menu in figure 6 has five items namely *New game*, *Load game*, *Game help*, *Options*, *Exit*. Function for menu control is shown as the source code in lines 72 to 173. Each menu item is represented by its own axes object. There are two pictures for each menu item, black for inactive state and grey for active item. The grey picture is shown when the item is selected or chosen. Moving up is realized by "w" key – line 75, down by "s" key – line 79 and selection by "l" key – line 83. You should be familiar with the basics of programming in Matlab to fully understand it. Let us describe the function in brief. All necessary variables are loaded before the function body using the line 73, file *defineglobal.m* Lines 74 to 105 show the menu control and selection. Line 74 shows how to load the key pressed – there is CurrentCharacter property for get function, and dMenu is the dialog identifier, lines 75 - 82 show moving up ad down. Lines 106 – 172 show the way of changing the pictures during move or selection. For one change of menu item all pictures are redrawn. This tip is based on the work of Hrubos consulted by me (Hrubos, 2009).

```
72    function keyboardmenu
73    defineglobal
74    switch lower(get(dMenu,'CurrentCharacter'))
75        case 'w'  % up
76            if kPosition <= 5 && kPosition > 1
77                    kPosition = kPosition - 1;
78            end
79        case 's'  % down
80            if kPosition < 5 && kPosition >= 1
81                    kPosition = kPosition + 1;
82            end
83        case 'l'
84            if kPosition == 1
85                    set(dMenu,'Visible','off');
86                    levelA
87            end
88            if kPosition == 2
89                    loadgame
90                    set(dLoad,'Visible','on');
91            end
92            if kPosition == 3
93                    gamehelp
94            end
95            if kPosition == 4
96                    set(dOptions,'Visible','on');
97            end
98            if kPosition == 5
99                    set(dMenu,'Visible','off');
100                   clear all
101                   exitmenu = 1;
102            end
103        otherwise
104            wavplay(zMenu,zMenuF)
105    end
106    if exitmenu == 1
107            clear all
108            return
109    else
110        set(dMenu, 'Units', 'pixels');
111        if kPosition == 1
112            set(0,'CurrentFigure',dMenu);
113            set(dMenu,'CurrentAxes',dMenuNewGameAxes);
114            image(oNewGame1); axis off;
115            set(dMenu,'CurrentAxes',dMenuLoadGameAxes);
116            image(oLoadGame0); axis off;
117            set(dMenu,'CurrentAxes',dMenuGameHelpAxes);
118            image(oGameHelp0); axis off;
119            set(dMenu,'CurrentAxes',dMenuOptionsAxes);
120            image(oOptions0); axis off;
121            set(dMenu,'CurrentAxes',dMenuExitAxes);
122            image(oExit0); axis off;
```

```matlab
123      elseif kPosition == 2
124         set(0,'CurrentFigure',dMenu);
125         set(dMenu,'CurrentAxes',dMenuNewGameAxes);
126         image(oNewGame0);axis off;
127         set(dMenu,'CurrentAxes',dMenuLoadGameAxes);
128         image(oLoadGame1);axis off;
129         set(dMenu,'CurrentAxes',dMenuGameHelpAxes);
130         image(oGameHelp0); axis off;
131         set(dMenu,'CurrentAxes',dMenuOptionsAxes);
132         image(oOptions0); axis off;
133         set(dMenu,'CurrentAxes',dMenuExitAxes);
134         image(oExit0); axis off;
135      elseif kPosition == 3
136         set(0,'CurrentFigure',dMenu);
137         set(dMenu,'CurrentAxes',dMenuNewGameAxes);
138         image(oNewGame0); axis off;
139         set(dMenu,'CurrentAxes',dMenuLoadGameAxes);
140         image(oLoadGame0); axis off;
141         set(dMenu,'CurrentAxes',dMenuGameHelpAxes);
142         image(oGameHelp1); axis off;
143         set(dMenu,'CurrentAxes',dMenuOptionsAxes);
144         image(oOptions0); axis off;
145         set(dMenu,'CurrentAxes',dMenuExitAxes);
146         image(oExit0); axis off;
147      elseif kPosition == 4
148         set(0,'CurrentFigure',dMenu);
149         set(dMenu,'CurrentAxes',dMenuNewGameAxes);
150         image(oNewGame0); axis off;
151         set(dMenu,'CurrentAxes',dMenuLoadGameAxes);
152         image(oLoadGame0); axis off;
153         set(dMenu,'CurrentAxes',dMenuGameHelpAxes);
154         image(oGameHelp0); axis off;
155         set(dMenu,'CurrentAxes',dMenuOptionsAxes);
156         image(oOptions1); axis off;
157         set(dMenu,'CurrentAxes',dMenuExitAxes);
158         image(oExit0); axis off;
159      elseif kPosition == 5
160         set(0,'CurrentFigure',dMenu);
161         set(dMenu,'CurrentAxes',dMenuNewGameAxes);
162         image(oNewGame0); axis off;
163         set(dMenu,'CurrentAxes',dMenuLoadGameAxes);
164         image(oLoadGame0); axis off;
165         set(dMenu,'CurrentAxes',dMenuGameHelpAxes);
166         image(oGameHelp0); axis off;
167         set(dMenu,'CurrentAxes',dMenuOptionsAxes);
168         image(oOptions0); axis off;
169         set(dMenu,'CurrentAxes',dMenuExitAxes);
170         image(oExit1); axis off;
171      else
172      end
173   end
```

Fig. 6. Part of dialog with set of "buttons" as pictures

8. Showing the vectors in dialogs

This part shows an example how to show vectors in dialog. There are several possibilities but this example seems to me as the simplest one. The realisation is provided in lines 174 to 251, and sample output is shown in figure 7. It is based on the following. The vectors of same length with numeric values change their data type using num2str function and they are added into one listbox in dialogue, each column for one data vector. And now the short description of provided source code is going to be outlined. Lines 174 and 175 predefine the name of colors to be used in the dialog, lines 176 - 180 defines new dialog, and lines 181 – 189 new menu and its items. The background of the dialog was created in Adobe Photoshop and the code for showing it in the dialog is in lines 190 – 192. There are several objects in the dialog. They are given by the code in lines 196 – 241. The position of objects depends on the screen resolution and therefore the k_y variable is used. The most important code for this example is in lines 242 – 251. The data are given via the String property, lines 250 and 251 have to be one line actually! The data type change is realized for the matrix, because each input in lines 250 and 251 is actually the vector.

```
174  cerna=[0 0 0];
175  cervena=[1 0 0];
176  hf6=figure('Color',cerna,...
177          'Name','Tabulka t, u1, y1, w1',...
178          'MenuBar','None',...
179          'Resize','off',...
```

```
180            'NumberTitle','off');
181  hmenu4=uimenu('label','Zobrazit tabulku :');
182  hmenu42=uimenu(hmenu4,'label','t,u2,y2,w2',...
183                 'CallBack','tu2y2w2');
184  hmenu43=uimenu(hmenu4,'label','t,e1,e2',...
185     'CallBack','te1e2');
186  hmenu44=uimenu(hmenu4,'label','t,T11,T21,K1',...
187     'CallBack','tT11T21K1');
188  hmenu45=uimenu(hmenu4,'label','t,T12,T22,K2',...
189     'CallBack','tT12T22K2');
190   axes('Position',[0 0 1 1]);
191   image(vysledky);
192   axis off;
193   close(hf10);
194   dial7;
195   pause(1);
196   text17 = uicontrol(hf6,...
197      'HorizontalAlignment','center',...
198      'BackgroundColor',cerna, ...
199      'ForegroundColor',cervena, ...
200      'Units','points', ...
201      'Position',[134 k_y*213.25 29.75 13.5], ...
202      'Style','text', ...
203      'String','t(s)');
204   text18 = uicontrol(hf6,...
205      'HorizontalAlignment','center',...
206      'BackgroundColor',cerna, ...
207      'ForegroundColor',cervena, ...
208      'Units','points', ...
209      'Position',[174 k_y*213.25 29.75 13.5], ...
210      'Style','text', ...
211      'String','u1');
212   text19 = uicontrol(hf6,...
213      'HorizontalAlignment','center',...
214      'BackgroundColor',cerna, ...
215      'ForegroundColor',cervena, ...
216      'Units','points', ...
217      'Position',[219 k_y*213.25 29.75 13.5], ...
218      'Style','text', ...
219      'String','y1');
220   text20 = uicontrol(hf6,...
221      'HorizontalAlignment','center',...
222      'BackgroundColor',cerna, ...
223      'ForegroundColor',cervena, ...
224      'Units','points', ...
225      'Position',[259 k_y*213.25 29.75 13.5], ...
226      'Style','text', ...
227      'String','w1');
228   text24 = uicontrol(hf6,...
229      'HorizontalAlignment','right',...
230      'BackgroundColor',cerna, ...
```

```
231        'ForegroundColor',cervena, ...
232        'Units','points', ...
233       'Position',[130 k_y*109.25 264.75 k_y*121.5], ...
234       'Style','frame');
235     text24 = uicontrol(hf6,...
236        'HorizontalAlignment','right',...
237        'BackgroundColor',cerna, ...
238        'ForegroundColor',cervena, ...
239        'Units','points', ...
240       'Position',[130 k_y*109.25 264.75 k_y*101.5], ...
241          'Style','frame');
242   ddd=size(param1); d1=ddd(1,2);
243     text24 = uicontrol(hf6,...
244          'HorizontalAlignment','right',...
245          'BackgroundColor',cerna, ...
246          'ForegroundColor',cervena, ...
247          'Units','points', ...
248         'Position',[134 k_y*113.25 256.75 k_y*93.5], ...
249          'Style','listbox', ...
250          'String',num2str([param1(:,d1) real(simout1(:,1))
251     real(simout1(:,2)) real(simout1(:,3))]));
```
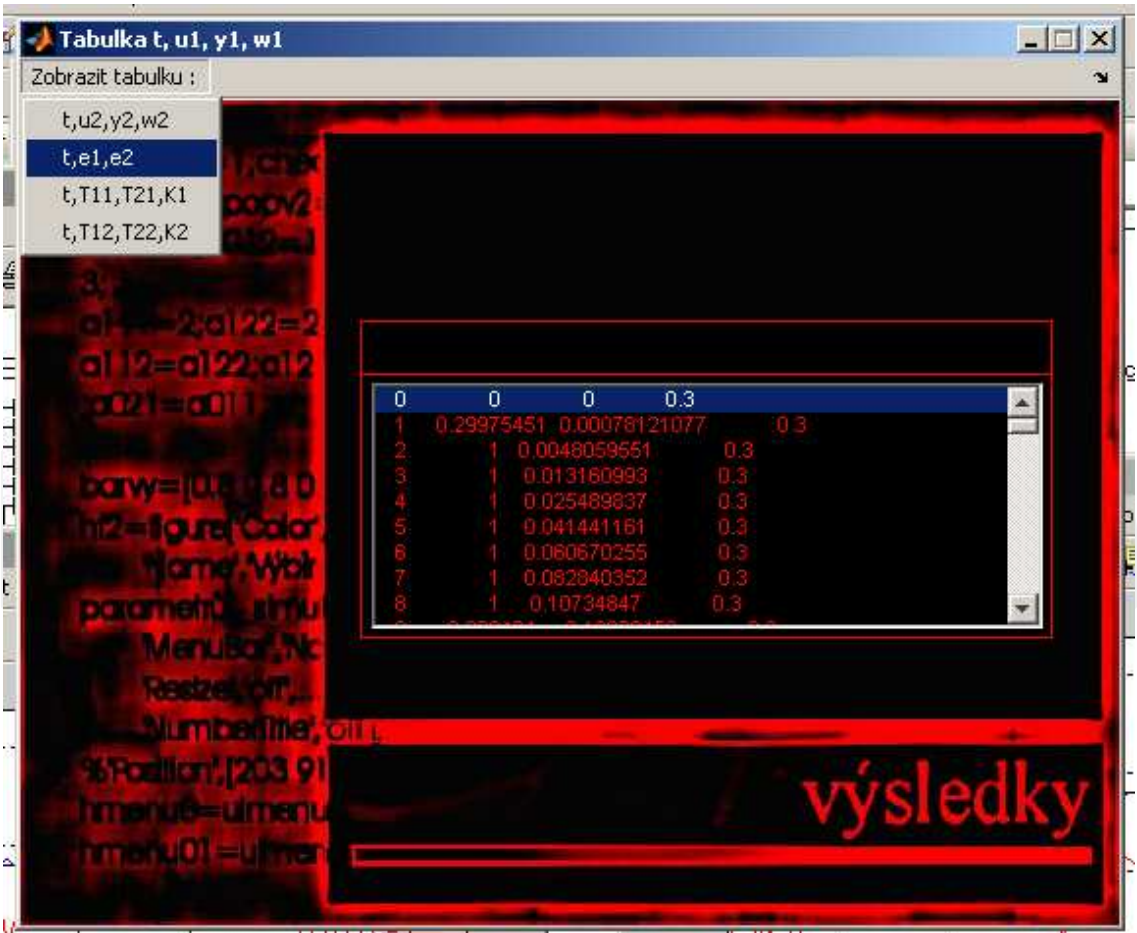

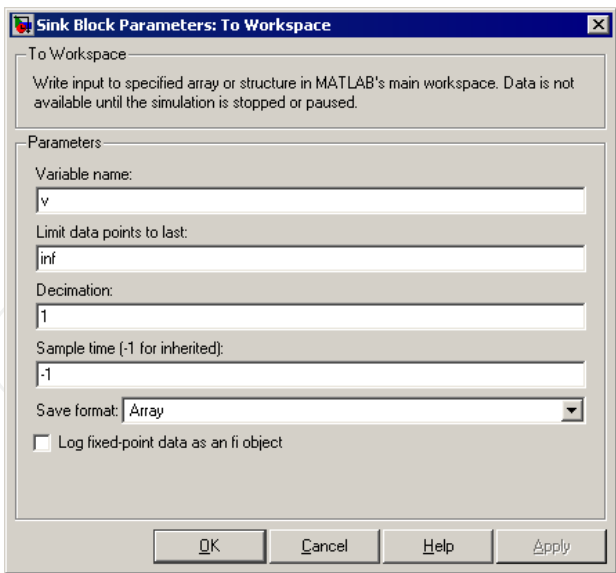Fig. 7. Dialog with vectors, data, shown in table

9. Nicely drawn results from Simulink

Simulink provides `Scope` block for drawing the results of simulation. It is a fast solution. If you want to manage the look of the graph, saving the data to Matlab Workspace and drawing them by the chosen plotting function is the easiest way. The following example shows it, lines 252 to 266 and figures 8 to 10.

```
252    close all
253    clear all
254    clc
255    a=0.1; b=0.01; c=0.5;
256    sim('untitled1',[0 10])
257    t=v(:,1);
258    x=v(:,2);
259    Dx=v(:,3);
260    plot(t,x,'r')
261    hold on
262    plot(t,Dx,'k')
263    title('x and its derivation')
264    xlabel('time [t]')
265    ylabel('x(t), Dx(t)')
266    legend('x(t)', 'Dx(t)')
```


Fig. 8. Simulink model with `Scope` and `To Workspace` blocks

Fig. 9. Setting the parameters of `Scope` block
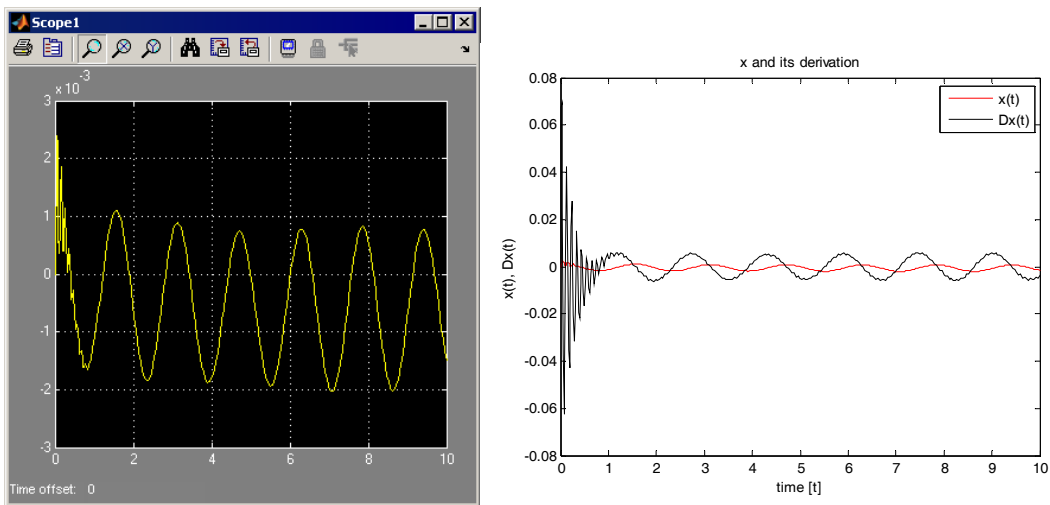
Fig. 10. Graphical output from Simulink (left) and using `plot` function (right)

Firstly, the Simulink model named as *untitled1.mdl* is created according to figure 8. `To Workspace` block saves the data into `v` variable as an array, see the settings in figure 9. Time is the first input to the `To Workspace` block. Simulation of the model is called by `sim` function, line 256. You can view the simulation results by clicking the blocks `Scope` and `Scope1` in the model, see figure 10 – left part. Lines 257 – 266 are responsible for nice and easy-to-edit graphical output, right part of figure 10. The data are sent to variables in lines 257 – 259.

10. Conclusion

This chapter presented some tips and tricks for programming in Matlab which appeared to be useful during the classes and it might be useful mostly for students and beginners. Almost all of them are oriented on working with GUI, only one tip is used with Simulink, the most common Matlab enlargement. The reader should be familiar with the basics of Matlab programming.

11. Acknowledgement

12. References

Hanselman, D.C. & Littlefield, B. (2005). *Mastering MATLAB 7*, Prentice Hall, ISBN 0-13-143018-1, USA.

Hrubos, P. (2009). *Software aid for Matlab teaching*, bachelor thesis, Tomas Bata University in Zlin, Zlin, Czech Republic (in Czech)

Perutka, K. (2005). *Matlab – Bases for students of automation and IT*, Tomas Bata University in Zlin, ISBN 80-7318-355-2, Zlin, Czech Republic (in Czech)

Perutka, K. (2007). *Decentralized adaptive control*, Ph.D. thesis, Tomas Bata University in Zlin, Zlin, Czech Republic, European Union (in Czech)

Perutka, K. & Heczko, M. (2007). Teaching of MATLAB Programming Using Complex Game. In: *FIE2007, 37th IEEE/ASEE Frontiers in Education Conference,* S1H 13-18, IEEE, ISBN 1-4244-1084-3, Milwaukee, WI, USA.
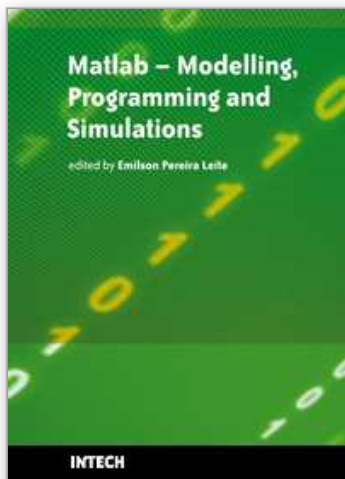
Matlab - Modelling, Programming and Simulations

Edited by Emilson Pereira Leite

This book is a collection of 19 excellent works presenting different applications of several MATLAB tools that can be used for educational, scientific and engineering purposes. Chapters include tips and tricks for programming and developing Graphical User Interfaces (GUIs), power system analysis, control systems design, system modelling and simulations, parallel processing, optimization, signal and image processing, finite different solutions, geosciences and portfolio insurance. Thus, readers from a range of professional fields will benefit from its content.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

INTECH
open science | open minds