8 Biomedical signal processing environment: Graphical User Interface

# 8.1 Introduction

A graphical user interface (GUI) is a user interface built with graphical objects; the components of the GUI; such as buttons, text fields, sliders, and menus. A well designed GUI is intuitively obvious to the user regarding the operation of the system. Applications that provide GUIs are generally easier to learn and use. The action that results from a particular user action can be made clear by design of the interface.

MATLAB implements GUIs as figure windows containing various styles of ui-control objects. We can program each object to perform the intended action when activated by user of the GUI. MATLAB's Graphical User Interface Development Environment (GUIDE) makes design easier.

A GUI can be created that sets the parameters of Simulink model. The Simulink diagram of the application must be open before running its GUI. If it is closed then GUI reopens it whenever it requires the model to execute. In addition, giving the facilities of obtaining simulation and saving the results and also plotting the results, the GUI can run the simulation and plot the results.

# 8.2 Implementation of GUI

Creating a GUI involves following basic tasks:

- ✓ Laying out the GUI components.
- ✓ Programming the GUI components.
- ✓ Saving and running the GUI.

GUIDE primarily is a set of layout tools and generates an M-file that contains code to handle the initialization and launching of the GUI. This M-file provides a framework for the implementation of the callbacks; the functions that execute when users activate components in the GUI. GUIDE stores GUIs in two files, which are generated the first time we save or run the GUI:

 FIG-file: A file with extension .fig that contains a complete description of the GUI figure layout and the components of the GUI: push buttons, menus, axes, etc. Changes to the GUI layout in the Layout Editor, our

Chapter 8: Biomedical signal processing environment: Graphical User Interface 198

changes are saved in the FIG-file.

M-file: A file with extension .m that contains the code that controls the GUI, including the callbacks for its components. This file is referred to as the GUI M-file. When we first run or save a GUI from the Layout Editor, GUIDE generates the GUI M-file with blank stubs for each of the callbacks. We can program the callbacks using the M-file editor.

Figure 8.1 illustrates the parts of GUI implementation:



Figure 8.1 Parts of GUI implementation

#### 8.3 GUIDE

It provides a set of tools for creating GUIs. These tools greatly simplify the process of laying out and programming a GUI.

When we open a GUI in GUIDE, it is displayed in the Layout Editor, which is the control panel for all of the GUIDE tools. The Layout Editor enables us to layout a GUI quickly and easily by dragging components, such as push buttons, pop-up menus, or axes, from the component palette into the layout area. The following picture shows the Layout Editor.



Chapter 8: Biomedical signal processing environment: Graphical User Interface 199

**Figure 8.2 Layout Editor** 

Once GUI is layout and each component's properties are set, using the tools in the Layout Editor, we can program the GUI with the M-file Editor. Finally, when we press the Run button on the toolbar, the functioning GUI appears outside the Layout Editor window. The features of GUIDE are:

- ✓ Laying Out GUIS: Using The Layout Editor we can add and arrange objects in the figure window.
- ✓ Aligning Components in the Layout Editor: We can align objects vertically or horizontally and can distribute them in group of components with respect to each other.

	Y Po PE AN AN IS DE DA RES OF	*
Nelect 🗧	📕 Align Objects	
Push But	Vertical	PushEutton
Togale 8	Align off Tr sta	Dunk Button
Radio Bu		
Chaskbar		Push Futton
Gileckou.	Met spacing 20 pixels	
IN EditText		Static Text
nor Static Teo	Horizontal	Popup Menu 💌
- Slider	Align OFF 📴 🛱 🛱	
Frame	Distribute	
Elistox	Bet spacing 20 pixels	
B Popup Mi	E set obtaining to	
IN Avec	DK Anoly Cancel	the second

Figure 8.3 Align objects

✓ Setting of Component Properties: Using The Property Inspector we can inspect and set property values. Property Inspector provides a list of all settable properties and displays the current value. Each property in the list is associated with an editing device that is appropriate for the values accepted by the particular property.



Figure 8.4 Property inspector

✓ Viewing the Object Hierarchy: Using The Object Browser we can observe a hierarchical list of the Handle Graphics objects in the current MATLAB session.



Figure 8.5 Object Browser

✓ Creating Menus: Using The Menu Editor we can create a menu bar; menus displayed on the figure menu bar; or a context menu; menus that pop up when users' right-click on graphics objects; for any component in the layout.



Figure 8.6 Menu Editor

✓ Setting the Tab Order: The Tab Order Editor displays the components of the GUI in their current tab order. Using The Tab Order Editor we can change the order in which components are selected by tabbing.





#### 8.4 Programming GUI Components

The section describes the features for programming the GUI components.

#### 8.4.1 GUI M-file

The M-file generated by GUIDE controls the GUI and determines how it responds to a user's actions, such as pressing a push button or selecting a menu item. **Figure 8.8** depicts execution path of m-file.

The M-file contains all the code needed to run the GUI, including the callbacks for the GUI components. While GUIDE generates the framework for this M-file, we must program the callbacks to perform the desired functions. When GUI is run, the M-file creates a handles structure that contains all the data for GUI objects, such as controls, menus, and axes. The handles structure is passed as an input to each callback. These handles structure can be used to Share data between callbacks and access the GUI data. A code can also be added to:

- 1. Opening function; executes before the GUI becomes visible to the user
- 2. Output function; outputs data to the command line
- 3. Callbacks; so that it is executed each time the user activates corresponding component of the GUI.



Figure 8.8 M-file Execution path

#### 8.4.2 GUI Data Management

GUIDE provides a mechanism, called **the handles structure**, for storing and retrieving shared data using the same structure that contains the GUI component handles. The handles structure, which contains the handles of all the components in the GUI, is passed to each callback in the GUI M-file. Therefore, this structure is useful for saving any shared data.

- ✓ Designing for Cross-Platform Compatibility: We can use specific property settings to create a GUI that behaves more consistently when run on different platforms. e.g. use different the default font or the default background color or figure character units for different platforms.
- ✓ Types of Callbacks: We can define callbacks in addition to that defined by the uicontrol Callback property.
- ✓ Interrupting Executing Callbacks: By default, MATLAB allows an executing callback to be interrupted by subsequently invoked callbacks. The GUI programmer can control whether user actions can interrupt executing callbacks or not.

 Controlling Figure Window Behavior: When designing a GUI we need to consider how we want the figure window to behave. The user can appropriately control the behavior for a particular GUI depending on intended use.

After writing the callbacks, GUI can be executed by:

- 1. Selecting Run from the Tools menu or
- 2. Clicking the Run button on the GUIDE toolbar.

If we have not saved the GUI recently, GUIDE displays the dialog box shown in **Figure 8.9**.

GUIDE	×
?	Activating will save change <mark>s to your figure</mark> and M-file. Do you wish to continue?
	Do not show this dialog again.
	Yes No

Figure 8.9 Dialog box to save changes

Click Yes to save the GUI files. If the directory where we save the GUI is not on the MATLAB path, GUIDE opens the following dialog, giving us the option of changing the current working directory to the directory containing the GUI files, or adding that directory to the MATLAB path as shown in **Figure 8.10**. Click **OK** to change the current working directory, and then GUIDE opens the GUI.



Figure 8.10 Dialog box to change the current directory during execution

## 8.5 Design of GUI for Signal Processing

The proposed models and training algorithms described in chapters 5, 6 and 7 are implemented in MATLAB 6.5.

Operation	Network Configuration	Training Algorithm	Link File Name
Filtering	Hopfield	LS	hopfield_filt.m
	(LS)		hopfield_filt.fig
	Hopfield (RLS)	RLS	hopfield_filt_rls.m
			hopfield_filt_rls.fig
	MLP ANN	Back	filtering_bkp.m
		Propagation	filtering_bkp.fig
	RBF ANN	RBF	filtering_rbfnn.m
		Training	filtering_rbfnn.fig
Compression	MLP ANN	Back	signal_comp_bkp.m
		Propagation	signal_comp_bkp.fig
	VQ ANN	Competitive	compression_vqnn_diff.m
	(Sample	Learning	compression_vqnn_diff.fig
	Difference)		
	VQ ANN	Competitive	compression_vqnn_no_diff.m
	(Absolute	Learning	compression_vqnn_no_diff.fig
	Samples)		
	MLP ANN	Extended	signal_comp_ext_bkp.m
		Back	signal_comp_ext_bkp.fig
		Propagation	
Detection	VQ ANN	Competitive	detection_vqnn.m
		Learning	detection_vqnn.fig

Table 8.1 Detailed Summary of various files used

A figure and script file ( selection.fig / selection.m ) is developed. On execution of script file ( selection.m ) or a .fig file ( selection.fig ) through GUIDE will generate main screen shown in **Figure 8.11** which allows user to select the signal processing operation, network architecture as well as training algorithm. When any button is pressed corresponding link file is activated and executed.

Close

🥠 Selection				
Artificial Neural Networks based Biomedical Signal Processing				
Filtering	Signal Compression			
Hopfield NN:LS	MLP ANN			
Hopfield NN:RLS	VQ ANN Absolute Samples			
MLPANN	VQ ANN Sample Difference			
RBF ANN	Extended Back Propagation			
Signal Detection 7	Display Input			
VQ ANN	Training Input Testing Input			
Noise Signal				
_	Close			

Figure 8.11 Graphical User Interface: Main Screen for selection

## 8.5.1 Filtering

Three network models are proposed for filtering, viz. Hopfield ANN, MLP ANN, RBF ANN. Pressing Hopfield NN:LS or Hopfield NN:RLS button will generate GUI of Figure 8.12 and Figure 8.13 respectively.

+) HOPFIELD(LS) Filtering	HOPFIELD(RLS) Filtering	
Hopfield Neural Network (LS)	Hopfield Neural Network (RLS)	
r 10000 =	r 10000 <u>*</u>	
c 0.000 ±	c 0.0000 •	
k1 1	k1 1	
k2 1	k2 1	
Training	Training	
M (Number of Training sets) 1000 Train	M (Number of Training sets) 1000 Train	
Training Data from file ECG1	Training Data from file ECG1	
Testing	Testing	
TST (Number of test sets) 3000 Test	TST (Number of test sets) 3000 Test	
Testing Data from file ECG1 -	Testing Data from file ECG1 -	
Mean Square Error (Training) :	Mean Square Error (Training) :	
Mean Square Error (Testing) :	Mean Square Error (Testing) :	
Return	Return	





Both the GUIs are same except that they will link to the training algorithm specified by the user. The default network parameters: Resistance (r), Capacitance (c), Multipliers (k1, k2), Number of Training Sets, Number of Test Sets are specified. User my change their value if he wishes, by entering new data in the space provided in the GUI.

Two default ECG data files named as ECG1 and ECG2 are provided for training as Train well as testing. Once the user selects the file name, button is enabled. On Train network will be trained using selected algorithm. On completion pressing Test Test button will be activated as Test of training. . Pressing button testing will be done (Figure 8.14 and Figure 8.15). The waveforms during training as well as testing described in chapter 5 will be generated. The performance of the operation is measured in terms of Mean Square Error (MSE). The value of MSE during training and testing will be displayed on GUI screen. On Return , the GUI is closed and main GUI will appear. pressing

HOPFIELD(LS) Filtering	_ 🗆 X	4 HOPFIELD(RLS) Filtering		
Hopfield Neural Network (LS)		Hopfield Neur	al Network (RLS)	)
r 10000 =		r	10000 +	
c 0.000 +		c	0.0000 ÷	
k1 1		k1	1	
k2		k2	1	
Training		Tra	aining	
M (Number of Training sets) 1000 Training	in	M (Number of Training sets	s) 1000 Trai	n
Training Data from file ECG1 -		Training Data from file	CG1 -	
Testing			Testina	Constanting of
TST (Number of test sets) 3000 Test	it	TST (Number of test sets)	3000 Tes	L
Testing Data from file ECG1 -		Testing Data from file	CG1 -	
Mean Square Error (Training) :		Mean Square Error (Trainir	ng) :	
Mean Square Error (Testing) :		Mean Square Error (Testin	g) :	
Return			Return	
Figure 8.14 GUI for Filtering	using	Figure 8.15 GU	JI for Filtering	using
Hopfield NN(LS Algorith)	m)	Hopfield NN	N(RLS Algorith	nm)

(with <u>Test</u> activated)

Hopfield NN(RLS Algorithm) (with Test activated)

On pressing MLPANN, GUI of Figure 8.16 will appear. The default values of the parameters are provided. However, user may change these values. Number of input layer nodes is decided by phase difference between noise signal and noise signal mixed up with information signal.

MLPANN Filtering	RBFANN Filtering
Feed forward neural network	Radial Basis Function Neural
for ECG filtering	Network for ECG Filtering
eta 0.1 +	SC (Span of Transfer Funciton) 800 +
H (Number of Hidden nodes) 2 +	EG (Error Goal) 0.1 +
Time (Number of epoches)  Training  M (Number of Training sets)  Training  T	Training       M (Number of Training sets)     100       Training Data from file     ECG1
Testing TST (Number of test sets) 3000 Test Testing Data from file ECG1 -	Testing
Mean Square Error (Training) : 0.0266607	Mean Square Error (Training) :
Mean Square Error (Testing) : 0.130402	Mean Square Error (Testing) :

## Figure 8.16 GUI for Filtering using MLP ANN (Values of MSE displayed)

## Figure 8.17 GUI for Filtering using RBF ANN

On pressing **RBF ANN** in main GUI, **Figure 8.17** will appear. This also allows setting parameters, select filename for training / testing. It displays results as well as plots during training / testing. It is possible for the user to use ECG files other than default. He can select 'Other' from the items of the popup menu. On selecting 'Other', **Help** will be activated (**Figure 8.18**) On pressing **Help** user may enter his own file name (**Figure 8.19**) which can be used for training or testing as per the selection through the same help window.

刘 RBFANN Filtering	👍 Help
Radial Basis Function Neural Network for ECG Filtering	2 ECG records and 1 EMG (noise) record.
SC (Span of Transfer Funciton) 800 • EG (Error Goal) 0.1 •	These are 8-channel records of 2 perpheral and 6 precordial leads, taken with respect to the left leg electrode. The 12 standard leads can be obtained by transform equations.
Training       M (Number of Training sets)     100       Help       Training Data from file       ECG1       ECG2       TST (Number of test sets)       Testing Data from file       ECG1	<ol> <li>The sampling rate is 400 Hz, with a resolution         <ol> <li>4.88 mV/bit and epoch length of 8 s, 12-bits ADC record (1 byte - last significant bits; 1 byte (half full) - most significant bits)).</li> <li>The first 2048 bytes of the records are for             patient data (name, age, etc.), followed by the             initial leads L,R,C1,C2,C3,C4,C5,C6 serially             recorded (all of L, all of R.).</li> </ol> </li> </ol>
Mean Square Error (Training) : Mean Square Error (Testing) : <b>Return</b>	Write full path of the file here.

Figure 8.18 GUI for Filtering using RBF ANN ( 'Other' Selection )

Figure 8.19 GUI for Help to select other file and to carry out training

#### 8.5.2 Compression

The network models are provided with environment viz. MLP ANN and Vector Quantization ANN (VQ ANN). Pressing any one button on the screen will select the model and training algorithm and corresponding GUI will appear.

MLPANN for Signal Compression	_	
Backpropagation		
I (Number of input nodes)	6	
O (Number of output nodes)	6	
H (Number of Hidden nodes)	3	
Time (Number of time epoches)	5	
eta	0.1	+
Training	1-1-1	-
M (Number of Training sets) 500	Train	
Training Data from file ECG1		
Testing		
TST (Number of Test Sets) 50	Test	
Testing Data from file ECG1 -		
Mean Square Error (Training):		
Mean Square Error (Testing):		
Return		

Figure 8.20 GUI for Compression using MLP ANN (Back Propagation)



Figure 8.21 GUI for Compression using MLP ANN (Extended Back Propagation)

J VQANN based Signal Compression	_ 🗆 🗙	🚽 VQANN based Signal Compression	_ 🗆 ×
Absolute Samples		Sample Difference	
I (Number of Input nodes) H (Number of Hidden nodes) Time (Number of time epoches) eta	4 128 • 8 0.05	I (Number of Input nodes) H (Number of Hidden nodes) Time (Number of Time Epoches) eta Training	4 128 + 8 0.05 + -
M (Number of Training sets)  128  Training Data from file  ECC1	Train	M (Number of Training Sets) 128 Training Data from file ECG1 •	Train
Testing Data from file ECG1	Test	Testing TST (Number of Test Sets) 500 Testing Data from file ECG1 •	Test
Mean Square Error (Training): Mean Square Error (Testing): <b>Return</b>		Mean Square Error (Training): Mean Square Error (Testing): Return	



Figure 8.23 GUI for Compression using VQ ANN ( Sample Difference )

Like previous GUIs, the default parameters and files are provided. User may modify the values.

#### 8.5.3 Detection

For VQ ANN based detection, parameters like number of input nodes, time epochs and eta are selectable (Figure **8.24**). Here number of competing nodes is not kept selectable because generally number of competing nodes are kept same as number of training sets.

📣 VQANN based Signal detection	×		
Vector Quantization NN for Signal Detection			
l (Number of Input nodes)	40		
Time (Number of time epoches)	8		
etaTraining	0.05		
M (Number of Training sets) 200	) Train		
Training Data from file ECG1	•		
Testing			
TST (Number of Test Sets) 500	) Test		
Testing Data from file ECG1	•		
Number of QRX Complexes:			
Return			

Figure 8.24 GUI for Detection using VQ ANN