

Introduction:

## GUI's using Matlab

English version: February 2010.

Original version: November 2007.

**Luis Pinedo Caro**  
**E-mail: [bizantino2@hotmail.com](mailto:bizantino2@hotmail.com)**

## Index:

1. ¿What is a GUI? ¿Why are they useful?
2. Meeting GUIDE
3. Your first GUI, steps to follow

## Preface:

This document is intended as a guide to build graphical user interfaces using Matlab. The difficulty I found when I tried to look for something easy and suitable for a real beginner made me write these notes with the sole purpose of serving me as a reminder. This document is intentionally biased towards the field of my studies – economics- and so the examples explained in the last section has to do with economic-based problems. These notes are written assuming the reader has no previous knowledge about building GUIs with Matlab, though some basics regarding Matlab programming are required.

## **1-¿What is a GUI? ¿Why are they useful?**

First of all a GUI (Graphical User Interface) is a tool that allow us to interact with a previously built Matlab program in such a way that every final user can have a quick and interactive glance of the problem.

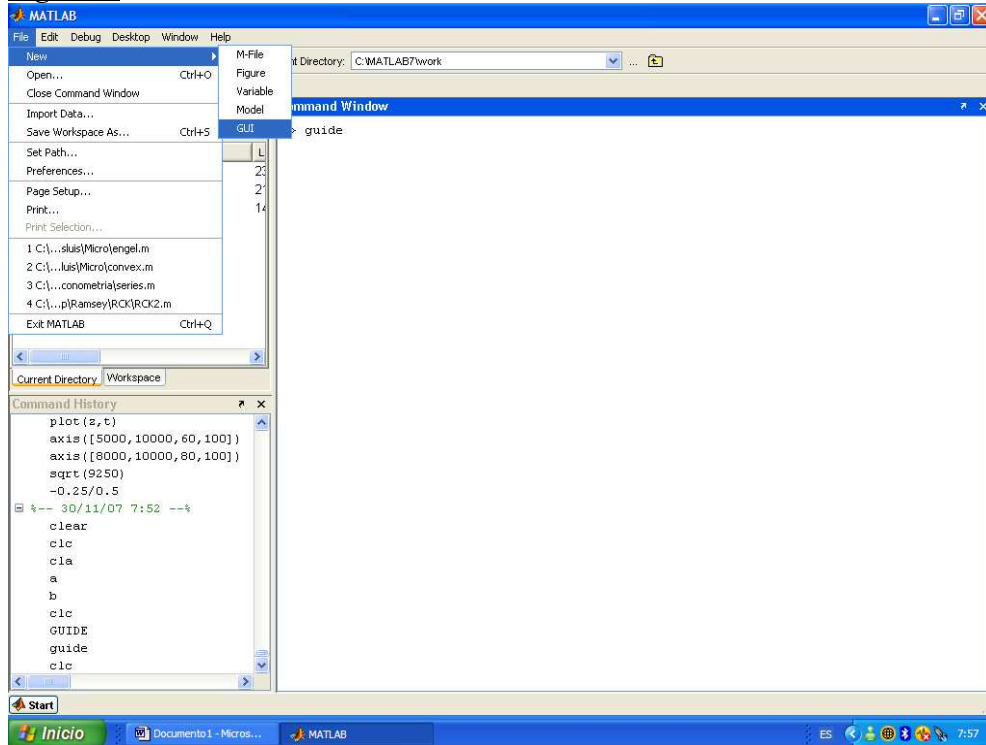
Secondly the necessity of building a GUI depends on each person and his needs. It is a fact that acquiring the skills and the knowledge has a high opportunity cost in terms of time so it is of no interest to fool anyone about the possibilities of such a tool. One of the advantages has to do with the ease of use. It might be difficult to share your Matlab programs with an inexperienced user. Also a figure or the Matlab output are sometimes not enough, either because you care about the procedure or because the final user will run the program several times. Here is where GUIDE appears as a powerful tool that provides your programs with ease of use and dynamism. Also GUIDE allows you to delimit the possibilities the final user will face, avoiding someone with little time or experience to be faced with the black box.

It is time to meet the application.

## 2-Conociendo la aplicación GUIDE

Let us open Matlab for a while. In order to open GUIDE you can either write in the Matlab command window `>> guide` or go through *File* → *New* → *GUI* as shown in figure 1.

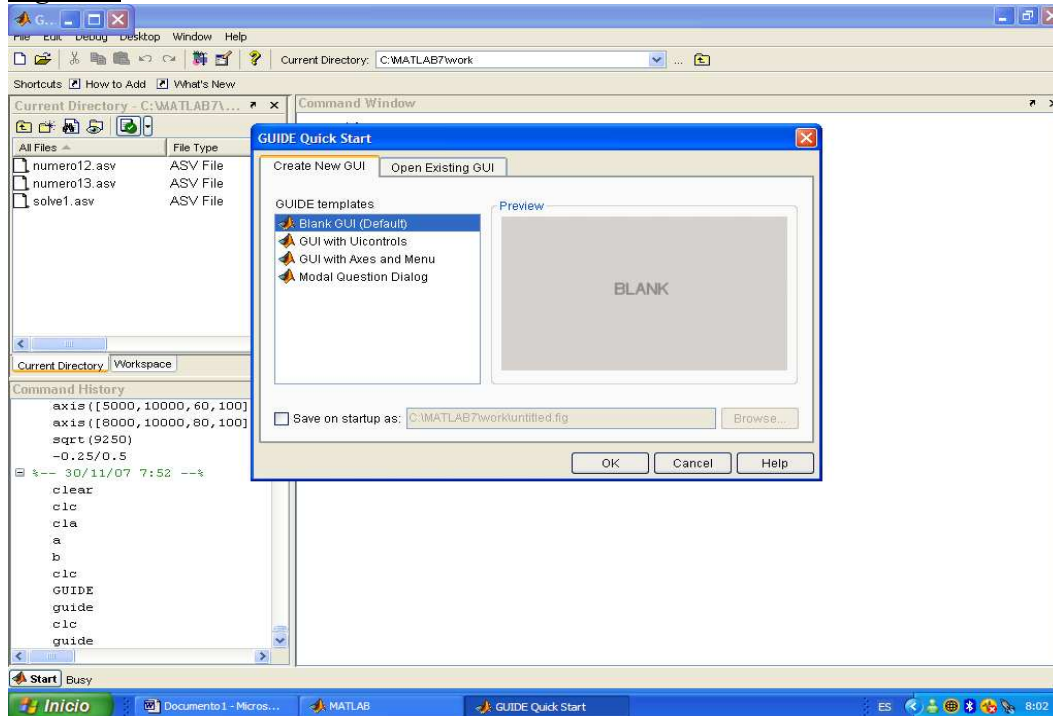
Figure.1



It will automatically pop up a small window with two tabs (Figure.2), the left one is intended to start a new GUI and the second one is to open an already created one. We can also see the option to create a predefined GUI with Uicontrols, axes or dialogs. We are not interested in the latter so let us click in Black Gui (default). Finally we can tick a box so as to save the .fig document before starting the GUI. That is irrelevant because it will not affect the design not the final result. Press ok and continue.

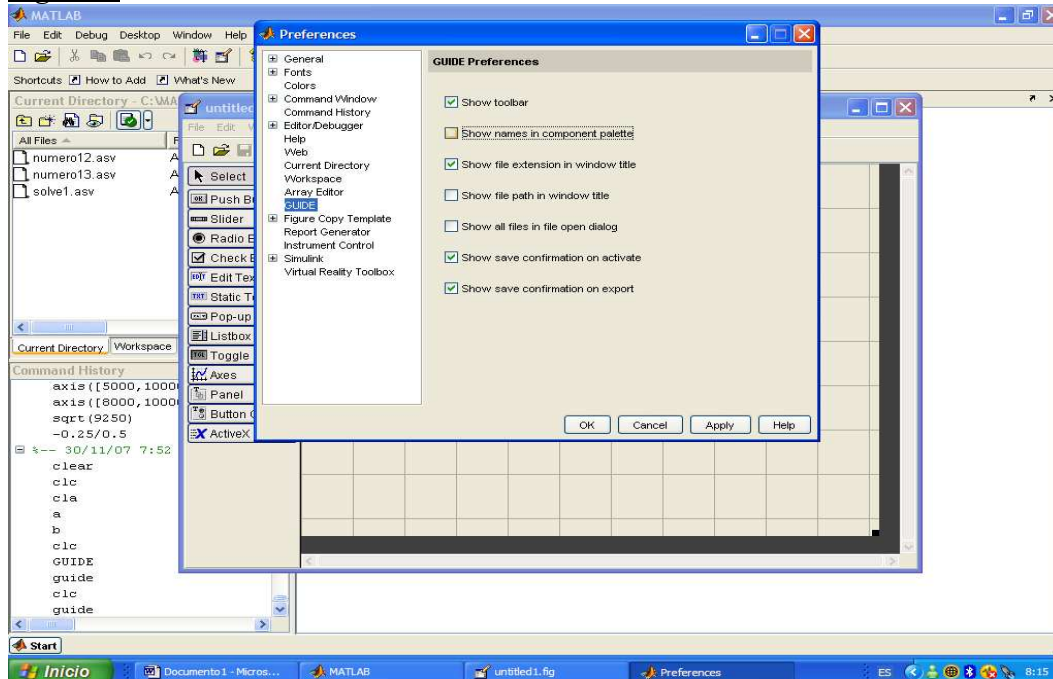
Now we can see our work environment. As it can be seen there are several buttons in the left hand side which are the add-ons we will use in our GUIs. At first sight it might be somewhat difficult to understand the meaning of every button so let us change this. Click on *File* → *Preferences* and now click on *GUIDE*; on the right we can see “show names in component palette” (Figure.3), choose it and accept. In order to add these components to our GUI we have to click and drag them to the interface area. It is advisable to click and drag several components to feel more comfortable with the buttons use. For instance Axis, Panel, Text and Push Button are among the most widely use. We will see how they work later.

Figura.2



Now we are about to see the most important feature when creating a GUI. It is to understand the correspondence between what we draw and the code Matlab generates at the same time.

Figure.3

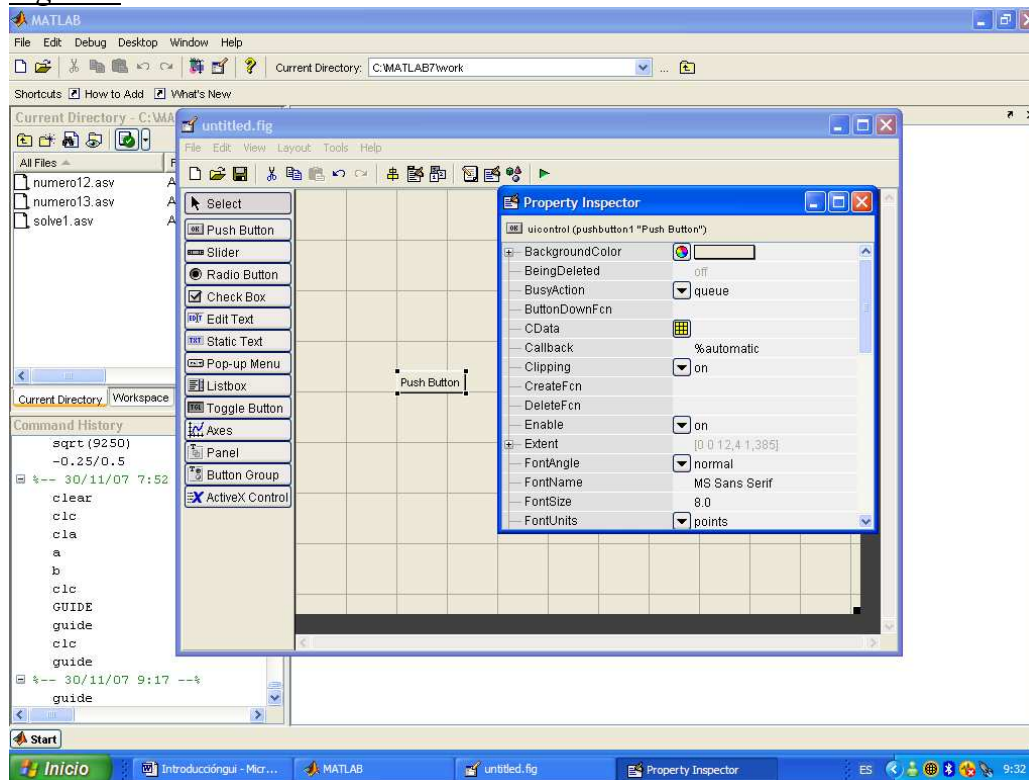


We must know that every GUI is made of two files, an m-file and a fig-file. Let us say the .fig file is the mask and the .m file is the machine that runs what we see on the mask. So far we are just giving shape to the mask, it is advisable though, to model the mask in a separate sheet before designing it using GUIDE. It would be useful to clarify our ideas

and more importantly, once the design has been saved Matlab creates the machine. The black box has everything needed to configure our current GUI but if we change the mask later because of a mistake Matlab may NOT change the black box to adapt it to the new changes so unless we know how to do so manually (and you will not in your first trials) get the final design at first.

Before jumping on to the black box configuration we are going to understand what we can do to set up the interface. We are going to introduce several examples, let us start clicking *Push Button* and dragging it on the design space as shown in figure 4. These buttons execute commands when pressed (i.e. run, help, exit, save). If we double click in the button we see the Property Inspector; before going mad because of the amount of options let us focus just on the most interesting ones. Firstly we see *Background color*, that is Matlab allow us to choose what color the button will show. There are two ways to change the color; click the color wheel to choose one predetermined color or in case you know the RGB colors just choose your preferred combination where there is a + sign.

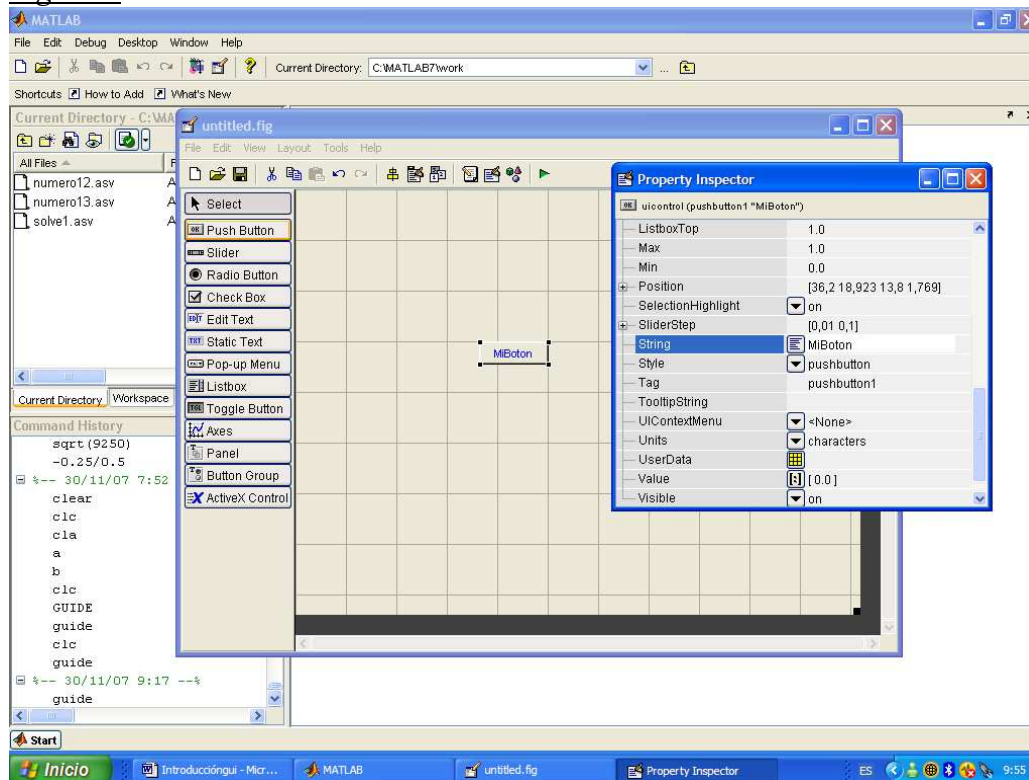
*Figure.4*



The next option I am going to explain is known as Callback, this property is the link between the visual interface and the machinery that is being run behind. As you can see it appears *%Automatic*, that is, once you have saved the GUI Matlab will create the callback structure automatically. We will see how to configure the callbacks later. Keep scrolling down until you see *Font Angle, Font Name, Font Units, Font Weight, Font Size and Foreground Color*, all of these properties have to do with the font configuration you can choose between normal font, bold or italic, size, color... Just change some of them to get a flavour of the possibilities. Next we are going to see two important options. On the one hand we see the option called *String* that is what we

see in the interface. For instance click on the empty space near String and write Mybutton, now check how the name of the button has changed.

*Figure.5*



On the other hand we have the Tag, this one is even more important than the former. It is name that Matlab gives to the callback. For example, if you write button1 Matlab will create button1\_callback. Why is it important? Just imagine a GUI with eight push buttons called button1, button2... button8, no one can remember the purpose of every button and it creates the chaos in your code –and remember, you have to find the callbacks in the .m file-. That is the reason why you should write a useful name that tells you now and in the future what is the purpose of such a callback. Let me clarify this issue with an example, imagine you have a push button that runs a program, if this were the case just write “run” which clearly tells you this button starts your program.

There are more buttons apart from the push buttons. They are described below:

Edit text: It builds a box that we can use either for an input or to receive an output from Matlab.

Panel: It helps us grouping the GUI components. It improves the quality and also make differences between groups of buttons (i.e. radio buttons) when using button groups ( a kind of panel).

Axes: It builds an axis where we can plot Matlab output.

Static text: It is use for improving the appearance of the GUI adding explanations and labels in the screen.



Radio button: It serves for creating a choice. When put within a panel the choice they mutually exclusive, that is, you can only choose one at the time.

Check box: Check boxes are small squares that can be ticked to apply or select a property or characteristic. They are not mutually exclusive.

Although every bottom has some particularities in its property inspector it is boring and wasteful to explain them in detail. Nonetheless we will see some differences in the examples. So far we just want to get in touch with the GUI environment.

### 3-Your first GUI, steps to follow.

Now it is time to start building our first GUI. It will be as simple as possible, but with enough options so as to get some benefit from it. The steps we have to accomplish and the needs we have to fulfill are:

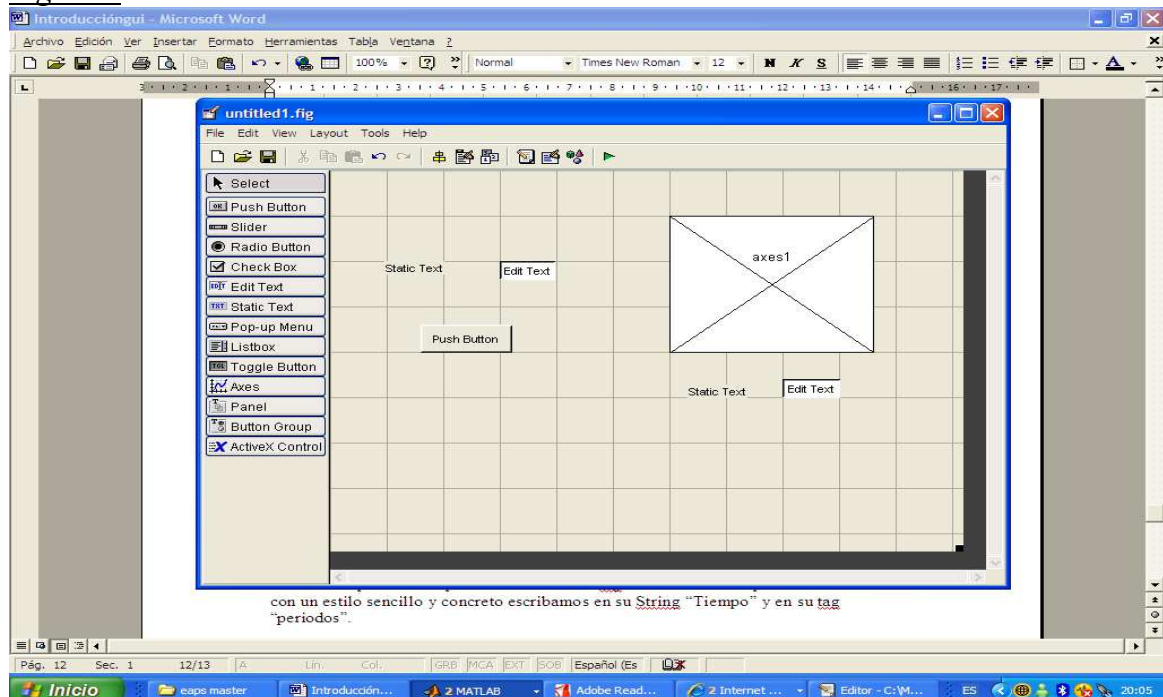
- 1°- Have a program that works.
- 2°- Have a design in mind.
- 3°- Set up the GUI's design.

Our first step is accomplished with a simple program that builds a numerical series around a given value. We build an interface that allows us to choose the length of the series and, at the same time, we ask Matlab about the process average. That way we are able to quickly see how the process tends to its average as the length tends to infinite. Of course the shock is distributed  $N(0,1)$ . The code looks as follows:

```
%Program that creates a numérica sequence around a given number.  
clear  
%Parameter  
T = 20;  
%Loop  
for t = 1:T  
    x(t)= 10+randn;  
end  
average=mean(x)  
plot(1:T,x,'k',1:T,average,'b:*')  
title('Numbers around 10')
```

This program works properly and we have already analyzed what we want from our interface, that is, a plot of the series, its average and the option to change the length of the process. Let us start the final step and so let us open Matlab. Type guide, open a new GUI and it is time to apply what we have learnt so far. We choose an axes for our plot, two edit texts (for the average and the length) two static texts (that will serve as tags for the latter characteristics) and a push button that will execute the program. Take a look at figure 6 to see a way of organizing the items.

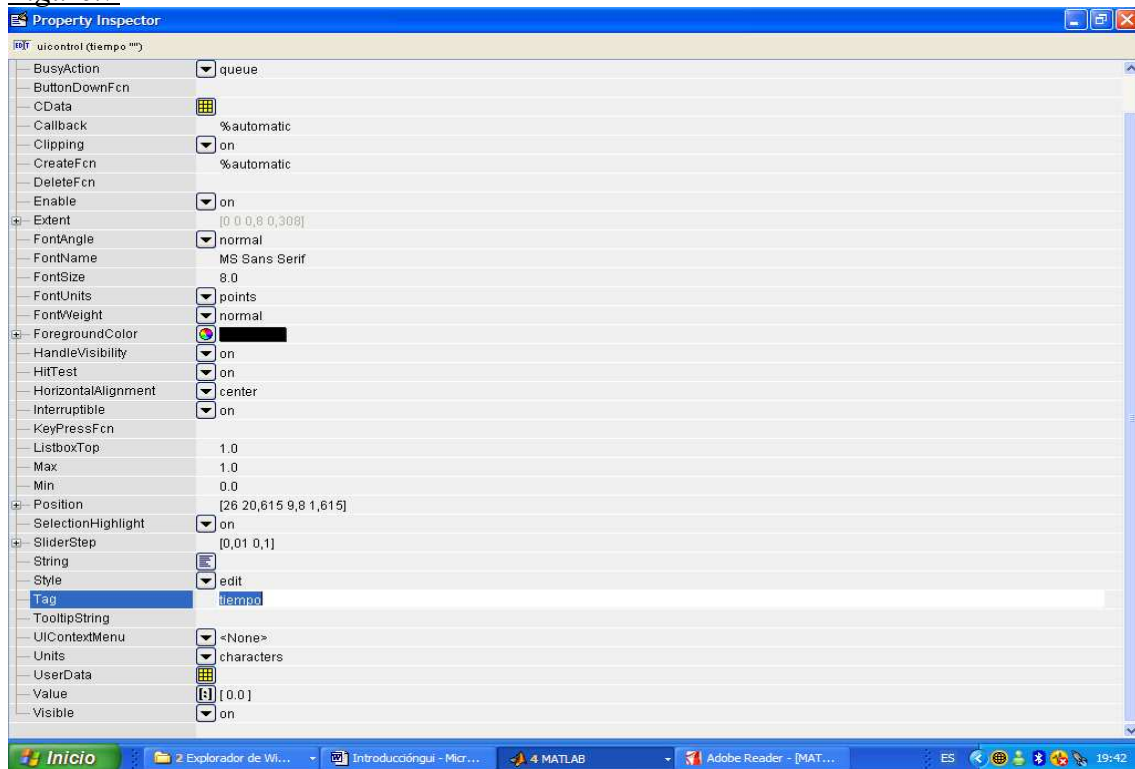
*Figure.6*



Next we prepare the items we have chosen. We start by double clicking on the left edit text. The property inspector will pop up. Go to string and leave blank; then go to the tag, which you know it is the callback name, so let us write something useful like “time” and press exe. We shall see something like figure 7.

Let us set up one of the static texts. As it is just a descriptive item we do not care too much about its tag. Nevertheless, in order to be coherent with the style we write “Periods” in its string and “periods” in its tag.

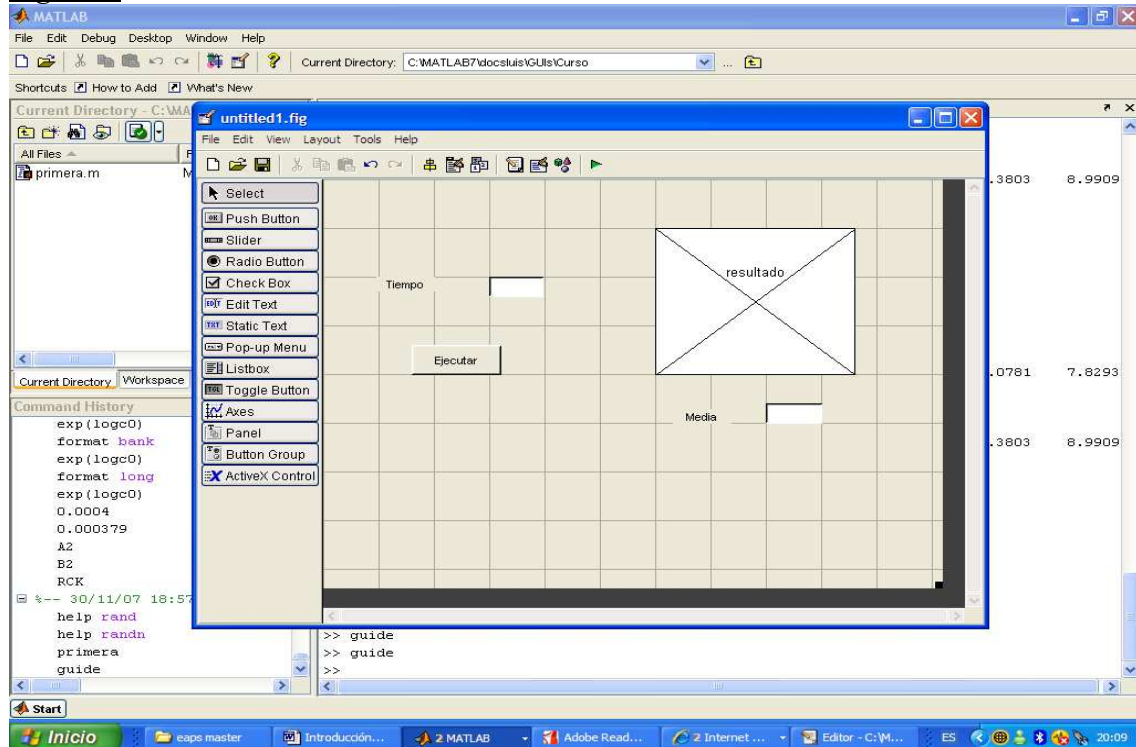
*Figure.7*



There is another edit text. This one will give us back the average calculated by Matlab. In this case it is compulsory to leave its string blank. The tag is called “average”. As we did before we name the static text’ string and tag Mean and mean respectively. Now write run and Run in the push button’ tag and string. Finally double click in the axes and write result in its tag. Out GUI should look like the one shown in figure 8.

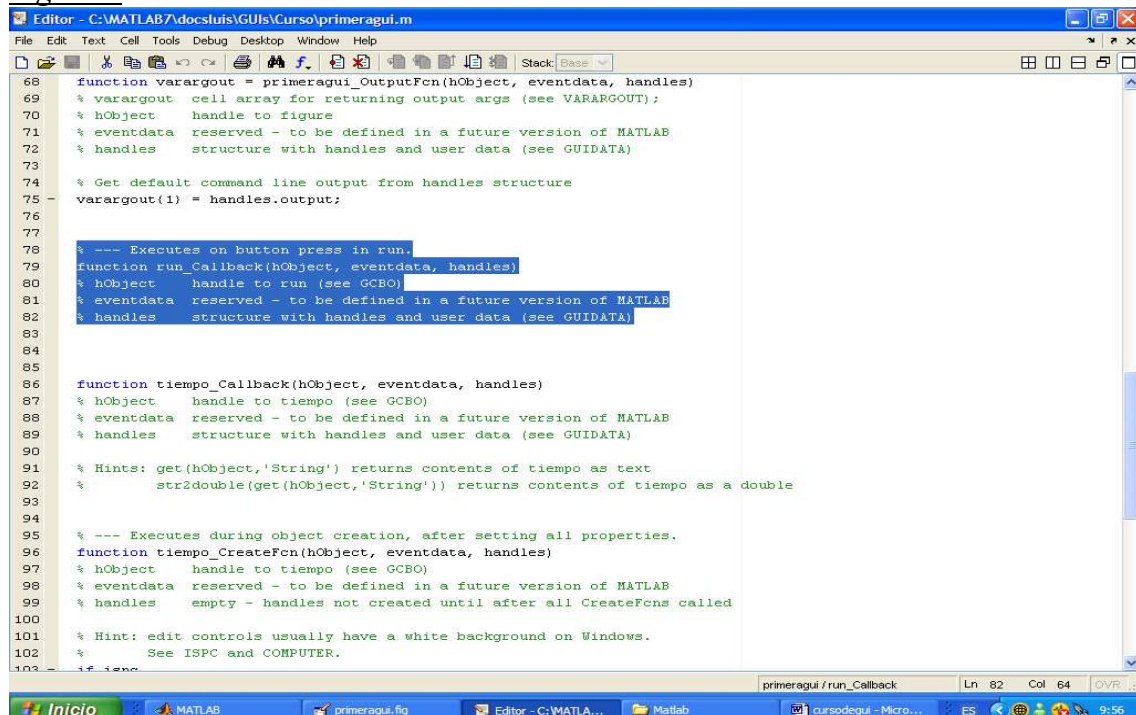
We have finished the GUI’s mask, so the first part of our task is done. In order to see how it looks we are going to save it. We will call it primeragui and Matlab will produce primeragui.m and primeragui.fig. You will notice that an m-file will pop up immediately so take the opportunity to say hello to the black box but don’t touch anything so far. Before properly setting it up let us click in the green triangle located in the above part of the m-file and you will how it looks –obviously nothing will happen as the interface does not have the code that makes it work-.

Figure.8



Now it is time to configure the black box created by Matlab. Its name (if you followed the instructions) is primeragui.m. First thing we you see the approximately 135 lines of code Matlab has created is not to run away. Second let us know where do we have to add some input for the GUI to be configured. Maybe you remind the word callbacks from above, which are the links between what we see and what Matlab executes. Well, these callbacks are exactly what we need to set up, so forget about every other stuff in the code. Figure 9 shows in blue the callback assigned to the pushbutton “run”.

Figura.9



We will write our bit just below the word function. Our bit is nothing else but our program's code. To sum up what we have done so far: We push the button Ejecutar, then Matlab through its callback looks up for the subfunction called run\_callback. Then Matlab finds some extra instructions written by us and simply it executes them. More questions; How does Matlab know how much is T? How does Matlab know where to plot our figure? These are things we will have to clarify to Matlab later. Now copy the program specified in figure.10, and never, ever copy the clear stuff—clear, clc, cla, ect—otherwise you will destroy your local variables and the program will not work.

Figura.10

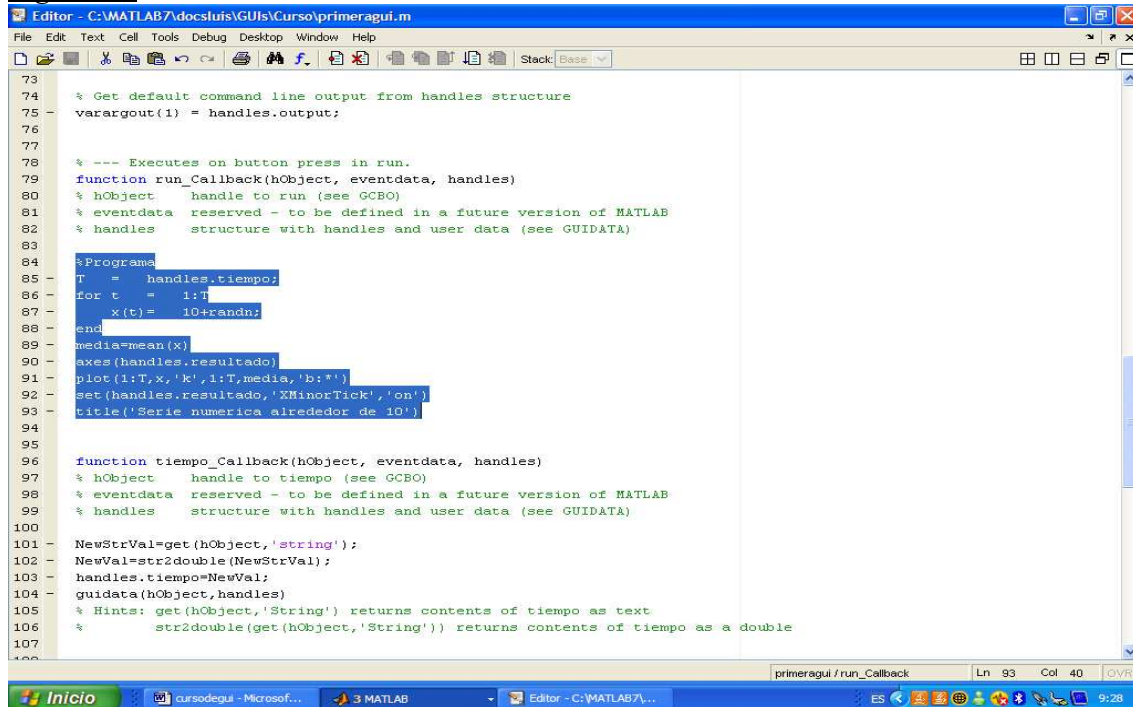
```

68 function varargout = primeragui_OutputFcn(hObject, eventdata, handles)
69 % varargout cell array for returning output args (see VARARGOUT);
70 % hObject handle to figure
71 % eventdata reserved - to be defined in a future version of MATLAB
72 % handles structure with handles and user data (see GUIDATA)
73
74 % Get default command line output from handles structure
75 varargout(1) = handles.output;
76
77
78 % --- Executes on button press in run.
79 function run_Callback(hObject, eventdata, handles)
80 % hObject handle to run (see GCBO)
81 % eventdata reserved - to be defined in a future version of MATLAB
82 % handles structure with handles and user data (see GUIDATA)
83
84 %Programa
85 T = 20;
86 for t = 1:T
87     x(t) = 10+randn;
88 end
89 media=mean(x)
90 plot(1:T,x,'k',1:T,media,'b:*)
91 title('Serie numerica alrededor de 10')
92
93
94 function tiempo_Callback(hObject, eventdata, handles)
95 % hObject handle to tiempo (see GCBO)
96 % eventdata reserved - to be defined in a future version of MATLAB
97 % handles structure with handles and user data (see GUIDATA)
98
99 % Hints: get(hObject,'String') returns contents of tiempo as text
100 % str2double(get(hObject,'String')) returns contents of tiempo as a double
101
102
103 % --- Executes during object creation, after setting all properties.

```

As a proof that we are actually giving instructions to Matlab save what you have done, run the program again (green triangle) and miraculously, your plot appears! Properly speaking this is more of an exception than the common rule. Let me explain this in further, when you only have a plot and an axis, Matlab will do automatically the assignment. But most times things are far from being that simple so let us write things properly. Add the following code just above plot: “axes(handles.resultado)”. This assigns a plot to an axis. We also add the following code right after plot: “set(handles.resultado,'XMinorTick','on’)”. Although this is not strictly necessary will help us knowing what Matlab is able to do. In this case the command set implement properties over an object. The property is XMinorTick and the state of the property is “on”. We can see the result in figure.11 underlined in blue.

Figura.11



```
73
74 % Get default command line output from handles structure
75 varargout(1) = handles.output;
76
77
78 % --- Executes on button press in run.
79 function run_Callback(hObject, eventdata, handles)
80 % hObject    handle to run (see GCBO)
81 % eventdata reserved - to be defined in a future version of MATLAB
82 % handles    structure with handles and user data (see GUIDATA)
83
84 %Programa
85 T = handles.tiempo;
86 for t = 1:T
87     x(t) = 10+randn;
88 end
89 media=mean(x)
90 axes(handles.resultado)
91 plot(1:T,x,'k',1:T,media,'b:*)
92 set(handles.resultado,'XMinorTick','on')
93 title('Serie numerica alrededor de 10')
94
95
96 function tiempo_Callback(hObject, eventdata, handles)
97 % hObject    handle to tiempo (see GCBO)
98 % eventdata reserved - to be defined in a future version of MATLAB
99 % handles    structure with handles and user data (see GUIDATA)
100
101 NewStrVal=get(hObject,'string');
102 NewVal=str2double(NewStrVal);
103 handles.tiempo=NewVal;
104 guidata(hObject,handles)
105 % Hints: get(hObject,'String') returns contents of tiempo as text
106 %       str2double(get(hObject,'String')) returns contents of tiempo as a double
107
108
```

Now we are going to set the callback specifying the number of periods. This is done as follows:

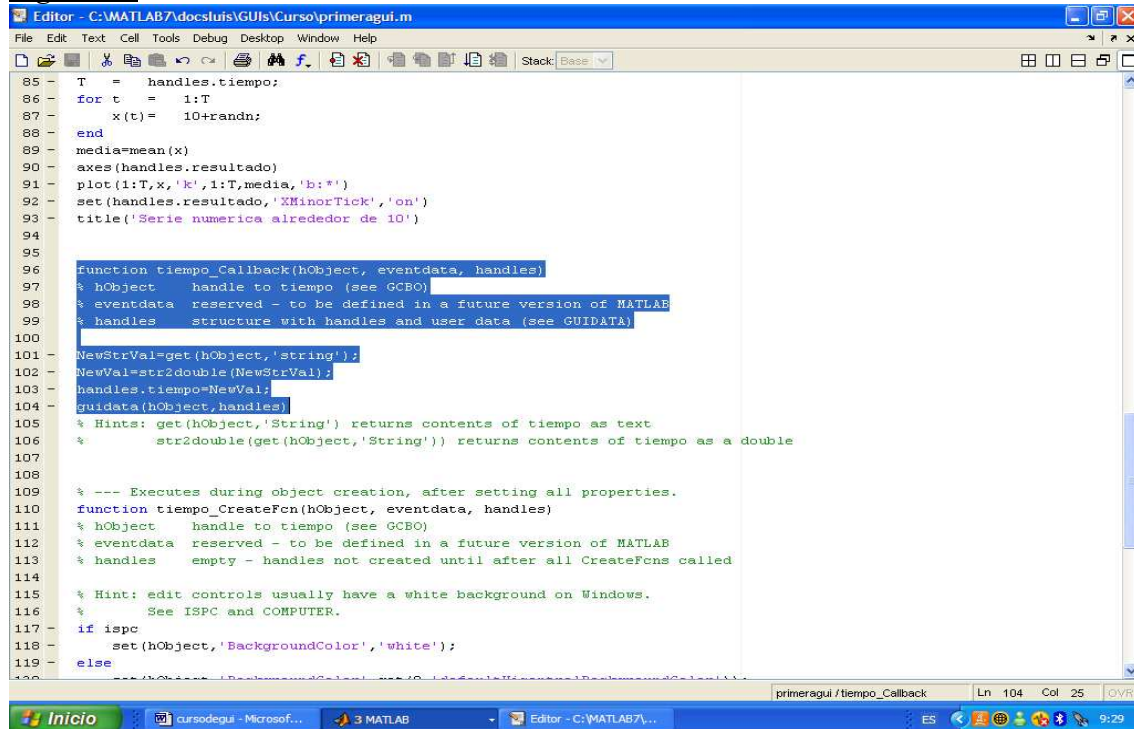
- 1° Capture the string of an edit text –in this case we use the edit text called tiempo-. An string might be the number we see in the interface.
- 2° Convert the string into a number –otherwise Matlab does look at it the same way we do-.
- 3° Storage the data and save the changes.
- 4° The program’s initial value is substituted by the newly created.

It is important to follow the steps carefully, at least the first time. Let us go to tiempo\_Callback (see figure.12) and write right after the function:

```
NewStrVal=get(hObject,'string');
NewVal=str2double(NewStrVal);
handles.tiempo=NewVal;
guidata(hObject,handles)
```

Let me explain you the code briefly; The first sentence is creating an object made of the string assigned to our edit text. The second sentence turns the object into a number (a double). The third part names the variable using handles, which is a reserved Word in Matlab and becomes part of the data. The last part stores the data for later use.

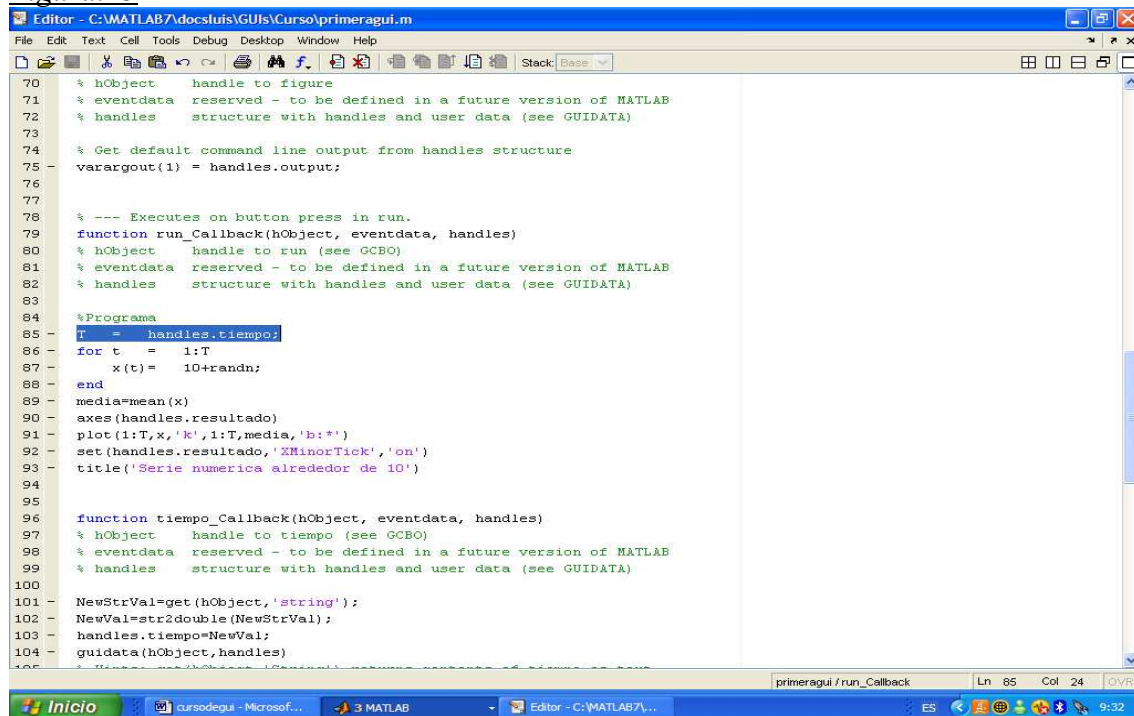
Figura.12



```
85 - T = handles.tiempo;
86 - for t = 1:T
87 -     x(t) = 10+randn;
88 - end
89 - media=mean(x)
90 - axes(handles.resultado)
91 - plot(1:T,x,'k',1:T,media,'b:*')
92 - set(handles.resultado,'XMinorTick','on')
93 - title('Serie numerica alrededor de 10')
94
95
96 function tiempo_Callback(hObject, eventdata, handles)
97 % hObject handle to tiempo (see GCBO)
98 % eventdata reserved - to be defined in a future version of MATLAB
99 % handles structure with handles and user data (see GUIDATA)
100
101 NewStrVal=get(hObject,'string');
102 NewVal=str2double(NewStrVal);
103 handles.tiempo=NewVal;
104 guidata(hObject,handles);
105
106 % Hints: get(hObject,'String') returns contents of tiempo as text
107 %       str2double(get(hObject,'String')) returns contents of tiempo as a double
108
109 % --- Executes during object creation, after setting all properties.
110 function tiempo_CreateFcn(hObject, eventdata, handles)
111 % hObject handle to tiempo (see GCBO)
112 % eventdata reserved - to be defined in a future version of MATLAB
113 % handles empty - handles not created until after all CreateFcns called
114
115 % Hint: edit controls usually have a white background on Windows.
116 %       See ISPC and COMPUTER.
117 if ispc
118     set(hObject,'BackgroundColor','white');
119 else
```

We have the value assigned in the black box but, does Matlab know where to use it? Not yet, so let us tell it. Go back to run\_callback (remember, our program), we see T=20, but we want it to have handles.tiempo. It is a piece of cake, substitute 20 for handles.tiempo and you are done. Check you have followed all the steps in figure.13.

Figura.13



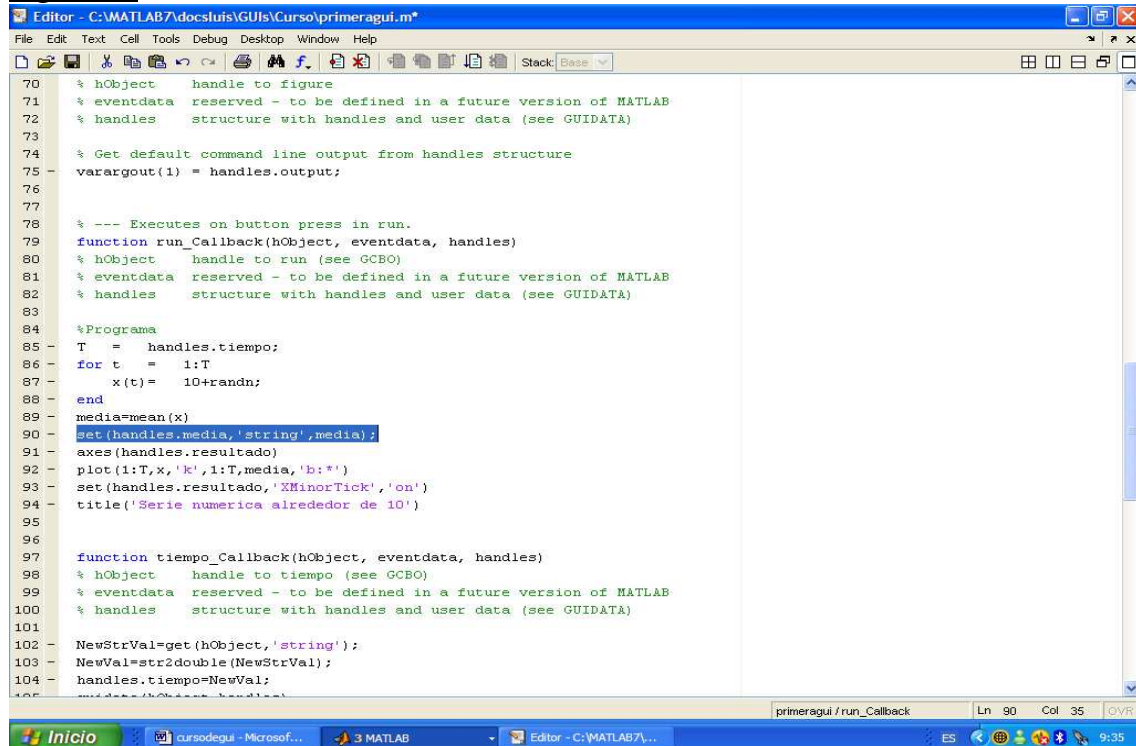
```
70 % hObject handle to figure
71 % eventdata reserved - to be defined in a future version of MATLAB
72 % handles structure with handles and user data (see GUIDATA)
73
74 % Get default command line output from handles structure
75 varargout(1) = handles.output;
76
77
78 % --- Executes on button press in run.
79 function run_Callback(hObject, eventdata, handles)
80 % hObject handle to run (see GCBO)
81 % eventdata reserved - to be defined in a future version of MATLAB
82 % handles structure with handles and user data (see GUIDATA)
83
84 %Programa
85 T = handles.tiempo;
86 for t = 1:T
87     x(t) = 10+randn;
88 end
89 media=mean(x)
90 axes(handles.resultado)
91 plot(1:T,x,'k',1:T,media,'b:*')
92 set(handles.resultado,'XMinorTick','on')
93 title('Serie numerica alrededor de 10')
94
95
96 function tiempo_Callback(hObject, eventdata, handles)
97 % hObject handle to tiempo (see GCBO)
98 % eventdata reserved - to be defined in a future version of MATLAB
99 % handles structure with handles and user data (see GUIDATA)
100
101 NewStrVal=get(hObject,'string');
102 NewVal=str2double(NewStrVal);
103 handles.tiempo=NewVal;
104 guidata(hObject,handles);
```

Here it comes the interesting part of this interface: the feedback. It is time to tell Matlab we want the average of the series. To do so we are going to use the command set (and we already know what it does) as follows:

```
set(handles.media,'string',media);
```

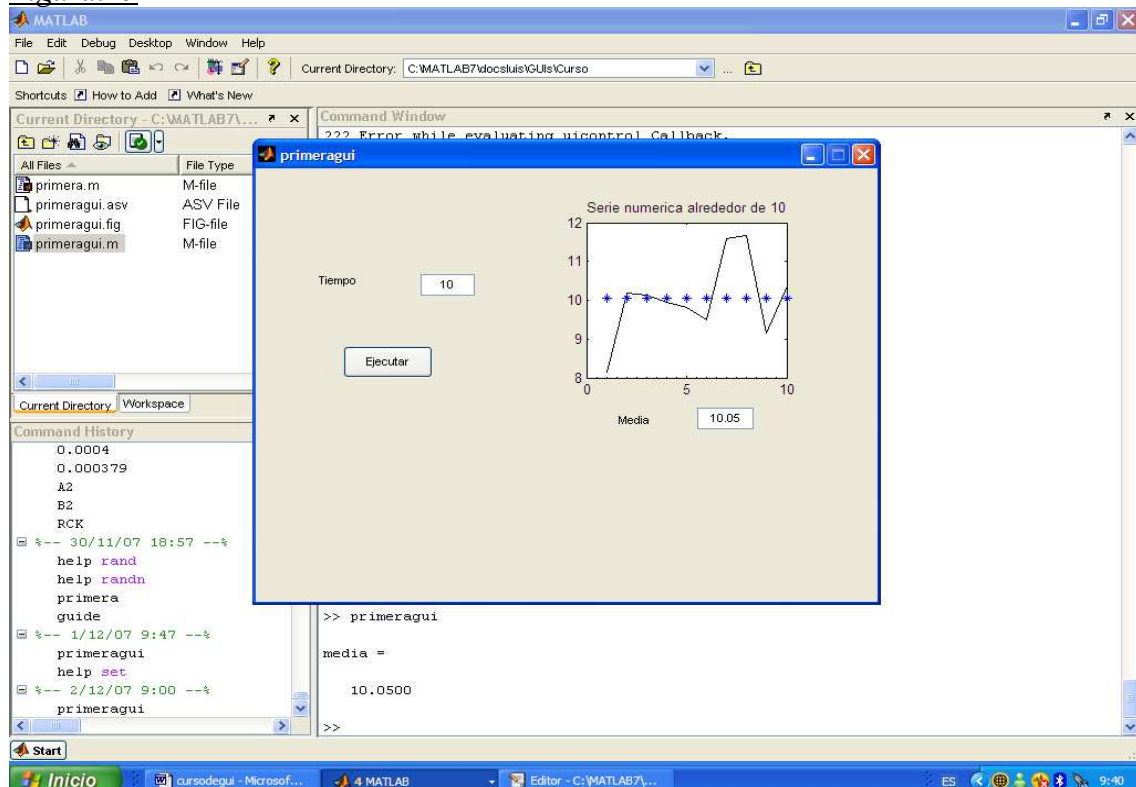
Remember, we put to assign –set- the string of handles.media (our text in the interface) to be the variable media obtained by Matlab.

Figura.14



```
70 % hObject handle to figure
71 % eventdata reserved - to be defined in a future version of MATLAB
72 % handles structure with handles and user data (see GUIDATA)
73
74 % Get default command line output from handles structure
75 varargout(1) = handles.output;
76
77
78 % --- Executes on button press in run.
79 function run_Callback(hObject, eventdata, handles)
80 % hObject handle to run (see GCBO)
81 % eventdata reserved - to be defined in a future version of MATLAB
82 % handles structure with handles and user data (see GUIDATA)
83
84 %Programa
85 T = handles.tiempo;
86 for t = 1:T
87     x(t)= 10+randn;
88 end
89 media=mean(x)
90 set(handles.media,'string',media);
91 axes(handles.resultado)
92 plot(1:T,x,'k',1:T,media,'b:*)
93 set(handles.resultado,'XMinorTick','on')
94 title('Serie numerica alrededor de 10')
95
96
97 function tiempo_Callback(hObject, eventdata, handles)
98 % hObject handle to tiempo (see GCBO)
99 % eventdata reserved - to be defined in a future version of MATLAB
100 % handles structure with handles and user data (see GUIDATA)
101
102 NewStrVal=get(hObject,'string');
103 NewVal=str2double(NewStrVal);
104 handles.tiempo=NewVal;
```

Figura.15





It is of good usage to write common objects as near as possible, so add the above sentence just below the variable `media` as shown in figure.14. Congratulations, you have finished your very first graphical user interface! Save the file, make sure both the `.fig` and the `.m` files are stored in the same folder, go to Matlab and call `primeragui`. The result should be similar to that shown in figure.15 where you may notice the average value of the process just in the empty space we left at the beginning.

Once you see the interface write a number in the space devoted to time (which should be appropriately signaled by a static text) and press `Ejecutar`. Surely it will not give you a prize but the hardest step has been done. You should have learnt what is a GUI, how to set it up, what is a callback, some properties of the GUI objects and a certain flavor about why a GUI might be of some use.