

Department of Electrical and Computer Engineering

EGE 331-Computer Simulation

Final Project

Design of a Graphical Signal Analyzer

|  |  |
| --- | --- |
| Date:05/07/2017 | Spring Semester 2017 |

|  |  |  |
| --- | --- | --- |
| Group Members | Department | Major Contribution |
| Joseph Abel | CE | Report/code |

Lecture Instructor: Professor Faramarz Vaziri

Abstract

    The objective of this project was to create a Graphical User Interface(GUI) which could graph any function in real-time and take the derivative and integral of it and also plot in real time. The Graphical Signal Analyzer had no constraints so the design was up to the user.

**Table of Contents**

Abstract……………………………………………………………………………………………………………………………………..1

Table Of Contents……………………………………………………………………………………………………………………..2

Introduction……………………………………………………………………………………………………………………………..3

Theory……………………………………………………………………………………………………………………………………..3

Design……………………………………………………………………………………………………………………………………3-5

Results and Conclusion……………………………………………………………………………………………………………....5

Appendix A………………………………………………………………………………………………………………………………..6-9

1. **Introduction**

Throughout the semester we have been learning the computer language MATLAB. MATLAB is a computer language written in C, C++ and Java and allows for numerical computing. In Computer Simulation we covered how MATLAB works and its different built in functions. In this project we will go through how to develop a GUI and how a Graphical Signal Analyzer works. A graphical signal analyzer reads an input signal and measures the power of the signal and displays it back out to you. In this project the Graphical Signal analyzer has to have a zoom in and out button, be able to move in different directions, add and remove grids and plot the derivative and integral.

1. **Theory**

In order to start with a MATLAB GUI you need to type guide in the MATLAB command interface. After this it will ask you if you want to create a new GUI. From here you will click yes and get sent to a blank screen which will allow you to create your GUI. In this screen you should be able to put down different items and tools which will allow you to interact with them. These tools can then be programmed using the MATLAB editor. There will be some preexisting code given to you but you need to fill in the rest with whatever functionality you need your GUI to do. The GUI has been around since the year computers have been invented they have allowed for people to enjoy computers through easiness. The users did not have to worry about interfacing with the command line. Every application in today’s society uses a GUI so these are good fundamental building blocks in how to program a GUI.

**III Design**

The idea behind how I developed the MATLAB GUI is to read in any function in real-time. In order to do this, I read in a function as a string and then get that function using a function called eval () this function takes a string forms an expression out of it. Next we had to take the derivative of that original function in order to do this we used a function called diff (). This function takes the derivative of the given function and displays it in the graph. The diff () function can also be used to take the difference between vectors in MATLAB. Lastly the function cumtrapz () computes an approximation of the cumulative integral via the trapezoidal method.

In order to graph these functions, we had to call each axes before we plotted it this is shown in figure 1.



Figure 1: Called axes than plots it

This similar procedure was followed for all three grpahs. Than next in order to turn every grid on at once we have button where every time the grid on button is pushed the grid was stay on this is illustrated in figure 2.

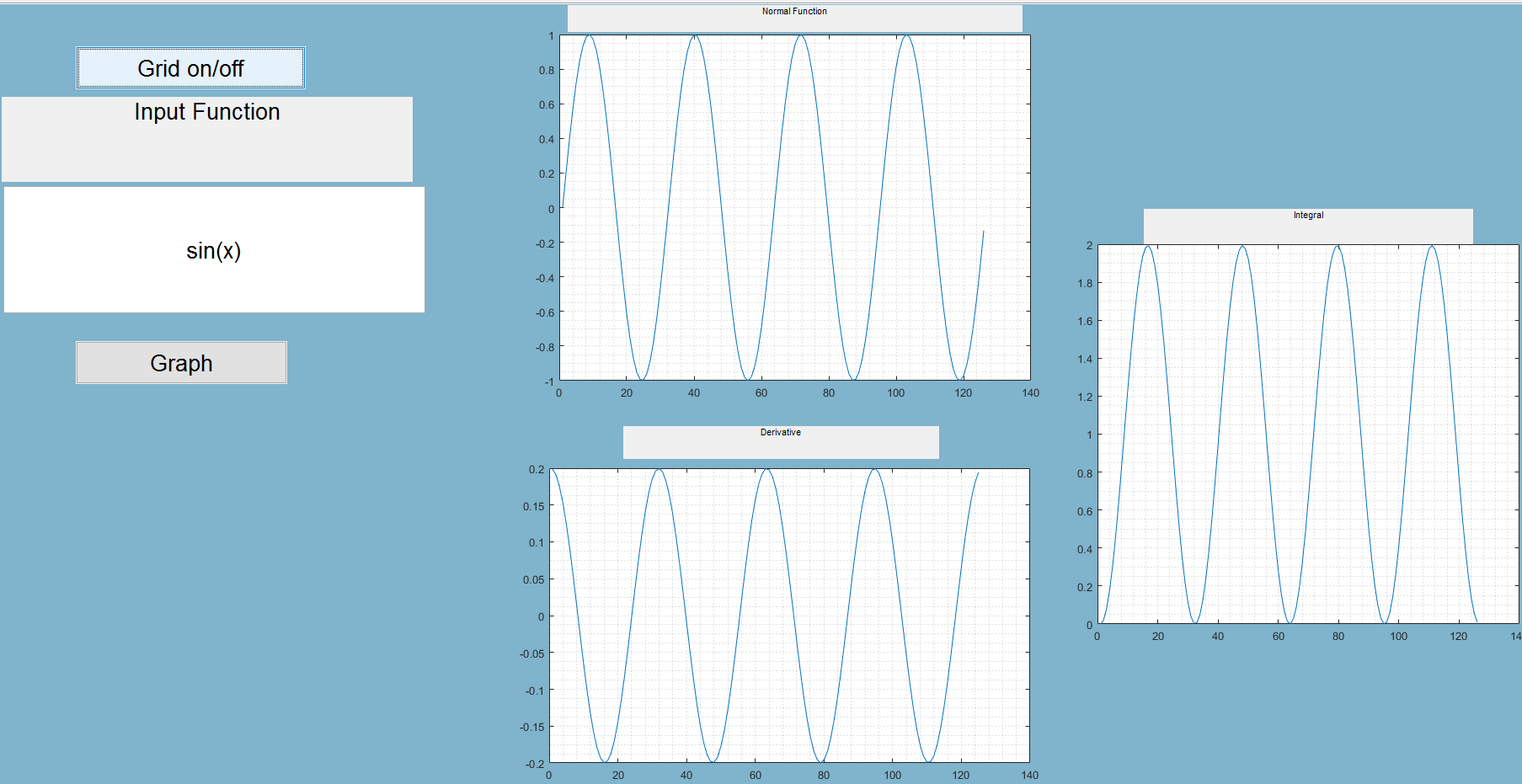


Figure 2: The entire MATLAB GUI

The next feature I want to show is the zoom in and out buttons. These buttons are from the MATLAB toolbox and they zoom in and out of my functions an example of this is shown in figure 3. As you will see in the top left the zoom in function is highlighted blue when in use.

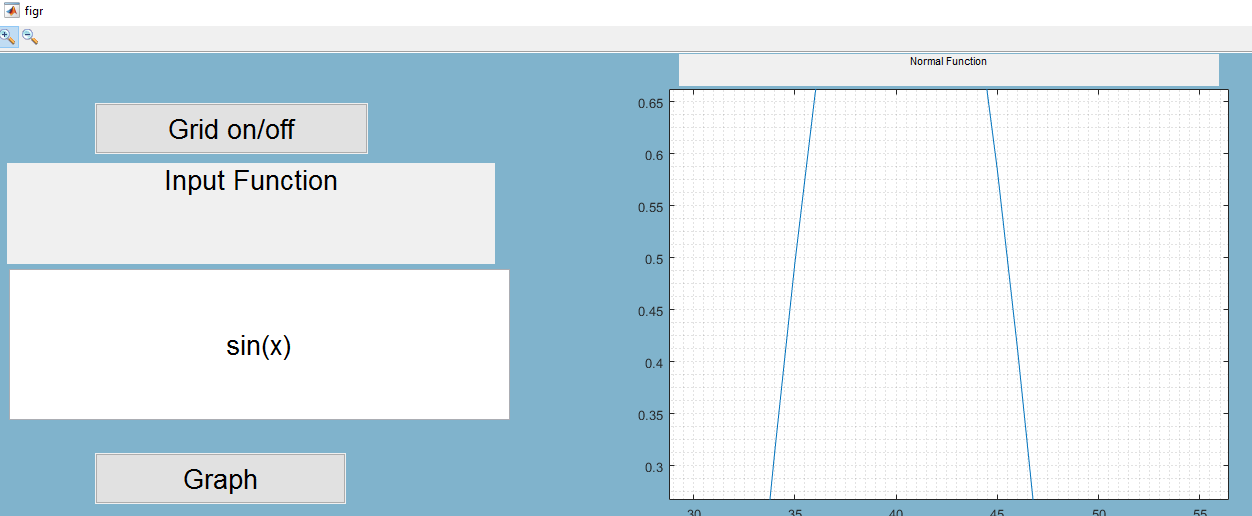


Figure 3: Zooming in on a sin(x) graph

1. **Results and Conclusion**

The results for this project were successful and using everything I learned from computer simulation. I was able to apply the concepts and create a real working Graphical User Interface. This GUI was able to plot any function for me in real time and give me the derivative and integral. This GUI was allowed me to be able to zoom in and out of certain functions as well as having a grid on or off for the graphs.

Appendix A MATLAB code

function varargout = figr(varargin)

% FIGR MATLAB code for figr.fig

% FIGR, by itself, creates a new FIGR or raises the existing

% singleton\*.

%

% H = FIGR returns the handle to a new FIGR or the handle to

% the existing singleton\*.

%

% FIGR('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in FIGR.M with the given input arguments.

%

% FIGR('Property','Value',...) creates a new FIGR or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before figr\_OpeningFcn gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to figr\_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 figr

% Last Modified by GUIDE v2.5 04-May-2017 13:59:16

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @figr\_OpeningFcn, ...

'gui\_OutputFcn', @figr\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(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 figr is made visible.

function figr\_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 figr (see VARARGIN)

% Choose default command line output for figr

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% UIWAIT makes figr wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = figr\_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;

function edit1\_Callback(hObject, eventdata, handles)

% hObject handle to edit1 (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 edit1 as text

% str2double(get(hObject,'String')) returns contents of edit1 as a double

% --- Executes during object creation, after setting all properties.

function edit1\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit1 (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 && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

%plot

% --- Executes on button press in pushbutton1.

function pushbutton1\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

x=-4\*pi:0.2:4\*pi;

%b=linspace(0,2\*pi)

a=get(handles.edit1,'string');

y=eval(a);

dy=diff(y);

axes(handles.axes1)

plot(eval(a))

axes(handles.axes2)

plot(dy)

Integral\_Of\_func=cumtrapz(x,y);

axes(handles.axes3)

plot(Integral\_Of\_func)

%grid on/off

% --- 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)

axes(handles.axes1)

grid minor

axes(handles.axes2)

grid minor

axes(handles.axes3)

grid minor

% --- 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

% --- 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.

if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor',[.9 .9 .9]);

end

% --- Executes on slider movement.

function slider2\_Callback(hObject, eventdata, handles)

% hObject handle to slider2 (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

% --- Executes during object creation, after setting all properties.

function slider2\_CreateFcn(hObject, eventdata, handles)

% hObject handle to slider2 (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.

if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor',[.9 .9 .9]);

end