An IDL Graphical User Interface for Performing a One Dimensional Fast Fourier Transform

María Helguera
Chaz Daly

Abstract

A GUI was designed using IDL's widget editor (IDL's command WIDED) to perform and display an FFT. Three windows allow the user to visualize his results. One window displays the signal in linear scale, while the other two display magnitude and phase or real and imaginary parts of the transform in either linear or dB scale. A choice of multiplying windows, Hamming, Hanning and Rectangular, is available. The user may select his choice by clicking on the appropriate buttons.

Introduction

We are interested in developing a graphical user interface (GUI) to perform a one dimensional FFT from a given real causal sequence. Our GUI is based on the program Signals developed by Prof. Roger Easton from the Center for Imaging Science at The Rochester Institute of Technology [2]. The GUI should be simple looking but appealing and easy to operate. To perform this task we decided to use the widget editor (WIDED) from IDL [1].

The difference between a GUI and a widget is hazzy. In a sense, a GUI is a generic concept, while a widget is a GUI written in IDL. For the purposes of this paper, GUI and widget are identical.

The user should not have to know anything about the underlying command structure of the code to use the widget program, although, as stressed later on, the input file must comply with a specific format.

A widget program responds to user inputs, such as clicking a button, or events generated in the program. This kind of program spends most of the time in an event loop. Once an event has occurred a module called event handler must process the event correctly.

The event is a message returned from the graphical interface as a result of an interaction of the user with the widget program. The information is packaged into an event structure that is passed to the event handler routine.
For each event handler in our program FFT_DIS.PRO a new widget structure is defined to place the data into the USER_VALUE widget structure defined in the top level base widget. The only communication between the widget definition module and the event handler module is via the event structure.

In widget programs, it is convenient to store several pieces of information in the user value of the top level base.

The user structure used by the event handlers in our program FFT_DIS.PRO is defined at the end of the code and described next for clarity:

graphs={sig:DRAW18_Id, ffta:DRAW19_Id, fftb:DRAW20_Id} assigns different windows for the signal and its transform display.

options={dB:0, ri:0} sets the defaults to linear and magnitude-phase.

widgets={data:FLTARR(1),win_data:FLTARR(1),no_elem:1, graphs:graph, options:options} takes care of format of data, number of elements, etc.

Most widget programs use the XManager procedure to take care of the event loop.

**Description of the FFT_DIS program**

In the description that follows, please refer to the commented code attached at the end of this paper.

The FFT widget has six buttons (three of them correspond to pull down menus) and three windows. The first button (pull down menu), DATA, selects the file to be transformed which can be read from memory or input manually. It has to be noted that the latter option has not yet been implemented.

To perform an FFT it is necessary first to read a file with the input data.

The input file must represent a real, causal signal and should have a header of two lines specifying in the first line the number of elements or data points in the array and in the second line a flag to let the widget know whether the file contains floating point values (0) or integer values (1).

This procedure can be seen in the event loop PRO PDMENU2_Event.

The FFT algorithm assumes that both the input function and the sampled spectrum are periodic over N samples [3]. If the input signal does not have an integral number of periods, “leakage” or “ringing” artifacts may occur in the spectrum. This artifact can be prevented if the input signal is forced to have a period of N samples by multiplying it with a smooth positive function (window) that decays to zero at the extremes of the interval (0,N-1). The spectrum of the
window will act as a lowpass filter; therefore, windowing will decrease leakage by sacrificing spectral resolution.

A choice of three windows is offered when pressing the second button (pull down menu), WINDOW, as can be seen in the event loop PRO PDMENU3_Event.

AHamming window is defined as
\[ w[n] = 0.54 + 0.46\cos\left(\frac{2\pi n}{N}\right) \]

A Hanning window is defined as
\[ w[n] = \frac{1}{2}\left(1 + \cos\left(\frac{2\pi n}{N}\right)\right) \]

A Rectangular window is defined as in [5] and is the default.

The default to display the transform is linear and magnitude-phase. Options for plotting dB scale and real and imaginary parts are offered when pressing the third button (pull down menu) PLOT OPTIONS. These can be seen in the event loop PRO PDMENU4_Event.

The actual computation of the FFT takes place in the event loop PRO MAIN13_Event. It is in this loop also that the plotting is done. A SHIFT is necessary to plot the FFT with the zero frequency component at the center of the display. Note that care has been taken to encompass both even and odd number of elements in the input file.

To calculate and plot the phase as \( \Phi = \text{atan}(\text{Im}(XX)/\text{Re}(XX)) \) it is necessary to extract the real and imaginary parts of the complex values of the transform. The real part is obtained by using the FLOAT function and the imaginary part by using the IMAGINARY function. Next, precautions are taken when either real or imaginary values (or both) are zero using the WHERE function in combination with a series of IF statements. The phase is calculated in degrees.

Combinations of magnitude-phase, real-imaginary, linear, dB can be chosen. However the option for real and imaginary parts in dB scale renders not very conventional results. To avoid taking the log of a negative number the absolute value is taken before plotting the log.

When clicking the fourth button, DISPLAY, the plots are displayed in their corresponding windows. This procedure can be seen in the event loop PRO MAIN13_Event under the label PDMENU4.

The fifth button is a HELP facility. Using the routine XDISPLAYFILE, a Readme file explaining how to use the widget is displayed on the screen.

Finally, the sixth button DONE, kills the widget when the user has finished its operation.
Results

Several test files were validated against results obtained with Signals [2] and Matlab [4]. These files were: a rect function, an exponential function $f[n] = \exp(-n/8)$, a sequence of integer values and a sequence of floating point values. Some of the results are attached.

Future directions and conclusions

Several things could be done to improve the current design:

Implement the manual entry for the input data.

Make the windows interactive so that the user could move a cursor to a given coordinate and be able to read the value at that point. This facility is provided in the program Signals.

Implement new buttons that allow the user to perform arithmetic operations among different files, i.e., multiply, exponentiate, convolve, etc.

As explained before, the option to display the real and imaginary parts of the transform in dB scale is not very orthodox. We were carried away with the different possibilities as a software problem but strictly speaking that option should be removed. In short, we were afflicted with “feature-itis”.

In summary, it was our purpose for this project to learn how to use the widget editor provided by IDL and to prove that a (relatively) simple problem, such as an FFT of a real causal signal, can be performed in a painless, and mostly transparent way for the user. To avoid extra complications we allowed the editor to take care of the aesthetics of the widget. That section of the code can be seen under the header “Widget definition built by Wided”.

We learned that widget programming is very different from the conventional top-down programming such as FORTRAN or Pascal.

Widget programming is an event based style of programming, similar to the window-based programming of Windows, Mac, and Xwindows. This style of programming lends itself to structures, pointing and indirection, i.e., to object oriented programming. This style makes widget programs easier to maintain and easier to modify and extend. It allows handling the data in a nondeterministic fashion. Even though we did not get fancy in our program we required a big structure to pass the data containing the user values.

Our experience showed us how difficult widget programming is. We hope people can learn from our widget and expand on it.
References


This widget reads in data from a file (manual entry is not working yet) and displays the one-dimensional FFT of the data. The display formats for the FFT are linear/dB, magnitude-phase/real-imaginary. The data can also be windowed with a Hamming, Hanning, or Rectangular window. The input data must be in a file in the following format:
Number_of_Elements  Float/Integer Data
where Number_of_Elements is an integer for the number of data points in the file
Float/Integer is a 0 for Floating point data and 1 for Integer data
Data is simply the floating point or integer data to FFT'ed

IMPORTANT: The Number_of_Elements and Float/Integer must be on separate lines in order for the widget to read the input data.
The input data must be real and is considered causal.

Default Display
Input data/Signal: always linear, cannot change
FFT: Linear magnitude, phase in degrees

Instructions:
Select the Data button to read in the data.
Select desired Window (Rectangular is default).
Select display Options (Linear, Magnitude/Phase is default).
Select Display to plot the data.
Select HELP for help.
Select DONE to quit.

Event Handlers with comments follow.

PRO PDMENU2_Event, Event ; Event handler for data entry

WIDGET_CONTROL, Event.top, GET_UVALUE = widgets

CASE Event.Value OF

'Data...File...': BEGIN ; Read data from a file
PRINT, 'Event for Data...File...'
filename = PICKFILE(/READ, 'FILTER = '*.dat')
OPENR, unit, filename, /GET_LUN
number_of_elements = 0
READF,unit, number_of_elements
is_data_int = 0
READF, unit, is_data_int
IF (is_data_int EQ 0) THEN BEGIN
data = FLTARR( number_of_elements )
ENDIF ELSE BEGIN
    data = INTARR(number_of_elements)
ELSE END
READF, unit, data
FREE_LUN, unit
END
'Data...Manual Entry...': BEGIN
    PRINT, 'Event for Data....Manual Entry...
END
ENDCASE

; Note that a new widget structure had to be defined in order to
; place the data into the USER_VALUE widget structure defined in
; the main widget. This is done for each event handler.

new_widget = {data: data, win_data: data, no_elem: number_of_elements,$
    graphs: widgets.graphs, options: widgets.options}
WIDGET_CONTROL, Event.top, SET_UVALUE=new_widget
END

PRO PDMENU3_Event, Event
    ;Window event handler
    WIDGET_CONTROL, Event.top, GET_UVALUE=widgets
    CASE Event.Value OF
        'Window....Hamming': BEGIN
            PRINT, 'Event for Window....Hamming'
            win_data = widgets.data * HANNING(widgets.no_elem, ALPHA=0.54)
        END
        'Window....Hanning': BEGIN
            PRINT, 'Event for Window....Hanning'
            win_data = widgets.data * HANNING(widgets.no_elem)
        END
        'Window....Rectangular': BEGIN
            PRINT, 'Event for Window....Rectangular'
            win_data = widgets.data
        END
ENDCASE

new_widget = {data: widgets.data, win_data: win_data, no_elem: widgets.no_elem,$
    graphs: widgets.graphs, options: widgets.options}
WIDGET_CONTROL, Event.top, SET_UVALUE=new_widget
END

PRO PDMENU4_Event, Event
    ;Plot options event handler
    WIDGET_CONTROL, Event.top, GET_UVALUE=widgets
    CASE Event.Value OF
'Plot Options....dB': BEGIN
PRINT, 'Event for Plot Options....dB'
dB = 1
ri = widgets.options.ri
END

'Plot Options....Linear': BEGIN
PRINT, 'Event for Plot Options....Linear'
dB = 0
ri = widgets.options.ri
END

'Plot Options....Magnitude/Phase': BEGIN
PRINT, 'Event for Plot Options....Magnitude/Phase'
dB = widgets.options.db
ri = 0
END

'Plot Options....Real/Imaginary': BEGIN
PRINT, 'Event for Plot Options....Real/Imaginary'
dB = widgets.options.db
ri = 1
END
ENDCASE

new_options = {dB:dB, ri:ri}
new_widget = {data:widgets.data, win_data:widgets.win_data, no_elem:widgets.no_elem, graphs:widgets.graphs, options:new_options}

WIDGET_CONTROL, Event.top, SET_UVALUE=new_widget

END

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

PRO MAIN13_Event, Event ;Event handlers for Display, Help, and Done

WIDGET_CONTROL,Event.Id,GET_UVALUE=Ev
WIDGET_CONTROL,Event.top,GET_UVALUE=widgets

CASE Ev OF
 ; Event for Data
'PDMENU2': PDMENU2_Event, Event
; Event for Window
'PDMENU3': PDMENU3_Event, Event
; Event for Plot Options
'PDMENU4': PDMENU4_Event, Event
'BUTTON5': BEGIN
    Print, 'Event for DISPLAY'
    XX = FFT(widgets.win_data)
    sig = widgets.graphs.sig
    ffta = widgets.graphs.fft
    fftb = widgets.graphs.fft
    dB = widgets.options.db
    ri = widgets.options.ri
    no_elem = widgets.no_elem
    even_odd = no_elem MOD 2
    IF (even_odd EQ 0) THEN BEGIN
        XX = SHIFT(XX, no_elem/2)
        bins = FINDGEN( no_elem ) - no_elem/2
    ENDIF ELSE BEGIN
        bins = FINDGEN( no_elem + 1 ) - no_elem/2
    END
XX = SHIFT(XX, FLOOR(no_elem/2))
ENDELSE
WSET, sig ;Plot FFT data
PLOT, widgets.data, YTITLE='Amplitude', XTITLE='Sample Number'
IF (dB EQ 0 AND ri EQ 0) THEN BEGIN
  WSET, ffta
  PLOT, bins, ABS(XX), YTITLE='Linear Magnitude', XTITLE='Frequency Bin'
  WSET, fftb
  phase = FLTARR(no_elem) ;Calculate phase
  real = FLOAT(XX)
  imag = IMAGINARY(XX)
  index = WHERE(imag EQ 0 AND real EQ 0, count)
  IF (count NE 0) THEN phase(index) = 0
  index = WHERE(imag EQ 0 AND real NE 0, count)
  IF (count NE 0) THEN phase(index) = 90
  index = WHERE(imag NE 0 AND real EQ 0, count)
  IF (count NE 0) THEN phase(index) = 90
  index = WHERE(imag NE 0 AND real NE 0, count)
  IF (count NE 0) THEN phase(index) = ATAN(imag(index), real(index))*(!RAD
  PLOT, bins, phase, YTITLE='Phase (degrees)', XTITLE='Frequency Bin'
ENDIF
IF (dB EQ 0 AND ri EQ 1) THEN BEGIN
  WSET, ffta
  PLOT, bins, XX, YTITLE='Real Part (Linear)', XTITLE='Frequency Bin'
  WSET, fftb
  PLOT, bins, IMAGINARY(XX), YTITLE='Imaginary Part (Linear)', XTITLE='Freq
ENDIF
IF (dB EQ 1 AND ri EQ 0) THEN BEGIN
  WSET, ffta
  magXX = ABS(XX) ;Find were FFT data is zero to prevent overflows
  index = WHERE(magXX NE 0, count)
  IF (count NE 0) THEN result = magXX(index)
  least = MIN(result)
  index = WHERE(magXX EQ 0,count)
  IF (count NE 0) THEN magXX(index) = least.
  PLOT, bins, magXX, YTITLE='dB Magnitude', XTITLE='Frequency Bin', /YLOG
  WSET, fftb
  phase = FLTARR(no_elem) ;Calculate phase
  real = FLOAT(XX)
  imag = IMAGINARY(XX)
  index = WHERE(imag EQ 0 AND real EQ 0, count)
  IF (count NE 0) THEN phase(index) = 0.0
  index = WHERE(imag EQ 0 AND real NE 0, count)
  IF (count NE 0) THEN phase(index) = 90.0
  index = WHERE(imag NE 0 AND real EQ 0, count)
  IF (count NE 0) THEN phase(index) = ATAN(imag(index),real(index))*(!RADE
  PLOT, bins, phase, YTITLE='Phase (degrees)', XTITLE='Frequency Bin'
ENDIF
IF (dB EQ 1 AND ri EQ 1) THEN BEGIN
  WSET, ffta
  real_mag = ABS(FLOAT(XX)) ;Prevent overflows
  index = WHERE(real_mag NE 0,count)
  IF (count NE 0) THEN result = real_mag(index)
  least = MIN(result)
  index = WHERE(real_mag EQ 0,count)
  IF (count NE 0) THEN real_mag(index) = least.
  PLOT, bins, real_mag, YTITLE='Real Part (dB)', XTITLE='Frequency Bin', /WSET, fftb
imag_mag = ABS(IMAGINARY(XX)) ; Prevent overflows
index = WHERE(imag_mag NE 0,count)
IF (count NE 0) THEN result = imag_mag(index)
least = MIN(result)
index = WHERE(imag_mag EQ 0,count)
IF (count NE 0) THEN imag_mag(index) = least
PLOT, bins, imag_mag, YTITLE='Real Part (dB)', XTITLE='Frequency Bin', /E
END

'BUTTON6': BEGIN
  Print, 'Event for Help'
  XDISPLAYFILE,'README'
END

'BUTTON7': BEGIN
  Print, 'Event for DONE'
  WIDGET_CONTROL, Event.top, /DESTROY
END

'DRAW18': BEGIN
  Print, 'Event for signal'
END

'DRAW19': BEGIN
  Print, 'Event for mag_real'
END

'DRAW20': BEGIN
  Print, 'Event for phase_imag'
END

ENDCASE
END

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
WIDGET DEFINITION built by WIDED
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

PRO fft_dis, GROUP=Group

IF N_ELEMENTS(Group) EQ 0 THEN GROUP=0

junk = { CW_PDMENU_S, flags:0, name:'' }

MAIN13 = WIDGET_BASE(GROUP_LEADER=Group, $
  COLUMN=2, $
  MAP=1, $
  TITLE='FFT_Display_Widget', $
  UVALUE='MAIN13')

MenuDesc122 = [ $
  { CW_PDMENU_S, 3, 'Data...' }, $ ; 0
  { CW_PDMENU_S, 0, 'File...' }, $ ; 1
  { CW_PDMENU_S, 2, 'Manual Entry...' } $ ; 2
]

PDMENU2 = CW_PDMENU( MAIN13, MenuDesc122, /RETURN_FULL_NAME, $
  UVALUE='PDMENU2')

MenuDesc124 = [ $
  { CW_PDMENU_S, 3, 'Window...' }, $ ; 0
{ CW_PDMENU_S, 0, 'Hamming' }, $ ; 1
{ CW_PDMENU_S, 0, 'Hanning' }, $ ; 2
{ CW_PDMENU_S, 2, 'Rectangular' } $ ; 3
]

PDMENU3 = CW_PDMENU( MAIN13, MenuDesc124, /RETURN_FULL_NAME, $
UVALUE='PDMENU3')

MenuDesc126 = [ $
{ CW_PDMENU_S, 3, 'Plot Options...' }, $ ; 0
{ CW_PDMENU_S, 0, 'dB' }, $ ; 1
{ CW_PDMENU_S, 0, 'Linear' }, $ ; 2
{ CW_PDMENU_S, 0, 'Magnitude/Phase' }, $ ; 3
{ CW_PDMENU_S, 2, 'Real/Imaginary' } $ ; 4
]

PDMENU4 = CW_PDMENU( MAIN13, MenuDesc126, /RETURN_FULL_NAME, $
UVALUE='PDMENU4')

BUTTON5 = WIDGET_BUTTON( MAIN13, $
UVALUE='BUTTON5', $
VALUE='DISPLAY')

BUTTON6 = WIDGET_BUTTON( MAIN13, $
UVALUE='BUTTON6', $
VALUE='Help')

BUTTON7 = WIDGET_BUTTON( MAIN13, $
UVALUE='BUTTON7', $
VALUE='DONE')

DRAW18 = WIDGET_DRAW( MAIN13, $
RETAIN=0, $
UVALUE='DRAW18', $
XSIZE=256, $
YSIZE=256)

DRAW19 = WIDGET_DRAW( MAIN13, $
RETAIN=0, $
UVALUE='DRAW19', $
XSIZE=256, $
YSIZE=256)

DRAW20 = WIDGET_DRAW( MAIN13, $
RETAIN=0, $
UVALUE='DRAW20', $
XSIZE=256, $
YSIZE=256)

WIDGET_CONTROL, MAIN13, /REALIZE

; Get drawable window index

COMMON DRAW18_Comm, DRAW18_Id
WIDGET_CONTROL, DRAW18, GET_VALUE=DRAW18_Id
; Get drawable window index

COMMON DRAW19_Comm, DRAW19_Id
WIDGET_CONTROL, DRAW19, GET_VALUE=DRAW19_Id

; Get drawable window index

COMMON DRAW20_Comm, DRAW20_Id
WIDGET_CONTROL, DRAW20, GET_VALUE=DRAW20_Id

; Define a user structure to be used by EVENT HANDLERS

graphs = {sig:DRAW18_Id, ffta:DRAW19_Id, fftb:DRAW20_Id}
options = {dB:0, ri:0}

widgets = {data:FLTARR(1), win_data:FLTARR(1), no_elem:1, graphs:graphs, optio

WIDGET_CONTROL, MAIN13, SET_UVALUE=widgets

XMANAGER, 'MAIN13', MAIN13
END