PHYS-4007/5007: Computational Physics

Using IDL in Workbench Mode

1 Starting IDL in Workbench Mode

There are two ways to run IDL under Linux: (1) the IDL Workbench Graphic User Interface (GUI) and (2) the IDL Command Line mode. This document will instruct you in this first method. First, log into your Linux user account on the TAF machines in Brown Hall 264 and open a Terminal window. Change directory to your idl subdirectory (*i.e.*, cd idl). Now start the IDL Workbench GUI by issuing the following command at the Linux prompt (indicated with a > sign below):

```
> idlde
```

It will take IDL a few seconds to get started the first time you run this software. The reason for this is that IDL will create a subdirectory in your login directory by the name of IDLWorkspace82. Once created, you are free to use this subdirectory as your working directory for IDL, however, I prefer to simply place all of my IDL files in a directory called idl in the login directory.

Note that you should get into the habit of storing all of the files you create on your account on a given machine to a USB Flash Drive in case you are unable to use the same machine the next time you wish to carry out computational work. Don't forget to Eject your USB Drive before logging off.

2 Editing a New IDL Procedure File

Select "New IDL Source File" on the "File" pull-down menu on the top right of the IDL GUI. This will bring up a text editing window in the GUI with the name "Untitled 1" on the upper-left tab of this text window. You will enter your code in this window.

Unlike Fortran 77, IDL is a *free-field* programming language – commands can start in any column on a line. Go to the "Useful IDL Procedures Web Page on the Course Web Page and click on the "mycode2.pro" link. Then copy and paste that code into the IDL GUI text editing box and select "Save to External Location ..." under the "File" pull-down menu to save this file to your workspace directory (make sure the file ends with the .pro filename suffix: mycode.pro). The actual code looks like this:

```
PRO MYCODE2
;
; This procedure draws a Gaussian and a circle and gives the student a
; lesson in making postscript (PS) hardcopy graphs.  Part of this code
; is used to draw Figure VI-1 in your notes.  This code is a bit more
; complex than the code shown in the IDL Command Line write-up.  This
; code can be retrieved from the Course Web Page.
;
TERM = !D.NAME  ; Get the name of the current device where graphs are drawn.
PASS = 0 ; Note that PASS=0 means we're plotting to terminal, =1 -> hardcopy
AFONT = [-1, 0]  ;  -1 -> default font for the terminal, 0 -> PS fonts
ASK = 'Y'  ;  Used in question read statements.
;
; The following are some common Greek letters and math symbol that can
; be used in the future should you ever need them.
;
INFINITY = ['!9$', '!7'+STRING(165b)]  ; Infinity sign [terminal, PS]
STHETA = ['!7h', '!7q']  &  SPHI = ['!7u', '!7f']
SCDELTA = ['!7D', '!7D']  &  SSIGMA = ['!7r', '!7s']
SMU = ['!7l', '!7m']  &  SCGAMMA = ['!7C', '!7G']
;
; Create x & y axes arrays to draw a circle.
;
ACIR = FINDGEN(361)/360  &  XCIR = COS(2*!PI*ACIR)  &  YCIR = SIN(2*!PI*ACIR)
;
; Create four arrays which will allow us to draw either horizontal or
; vertical curly braces.
;
YCURLY_X = [2.0, 1.5, 1.0, 0.5, 0.0, 0.0, 0.2, 0.4, 0.7, 1.0, 1.4, 1.8, 1.2, $
        1.8, 1.4, 1.0, 0.7, 0.4, 0.2, 0.0, 0.0, 0.5, 1.0, 1.5, 2.0]
YCURLY_Y = [0.0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4, $
        2.6, 2.8, 3.0, 3.2, 3.4, 3.6, 3.8, 4.0, 4.2, 4.4, 4.6, 4.8]
XCURLY_X = YCURLY_Y  &  XCURLY_Y = YCURLY_X
;
; Useful commands for draw big points when PSYM=8 in the PLOT command.
;
APT = FINDGEN(16)*2.*!PI/16.
USERSYM, 1.5*COS(APT), 1.5*SIN(APT), /FILL
;
; Set box characteristics
;
!X.THICK = 2.4  &  !Y.THICK = 2.4  &  !P.THICK = 2.4
```

```
!P.CHARSIZE = 1.5  &  !X.CHARSIZE = 1.0  &  !Y.CHARSIZE = 1.0
;
; Create x & y axes arrays containing a Gaussian function.
;
SIGMA1 = 10.  &  PE1 = 0.6745 * SIGMA1
X1 = DINDGEN(800)/10 - 39.9D0
Y1 = (10.0D0/SIGMA1/SQRT(2*!DPI)) * EXP(-0.5 * (X1/SIGMA1)^2)
YSIG1 = (10.0D0/SIGMA1/SQRT(2*!DPI)) * EXP(-0.5)
YPE1 = (10.0D0/SIGMA1/SQRT(2*!DPI)) * EXP(-0.5 * (0.6745)^2)
;
PLOTAGAIN:
;
; Draw the Gaussian.  Note that XSTYLE and YSTYLE keywords determines how
; axes drawn.  Here we are suppressing all axes.
;
PLOT, X1, Y1, XSTYLE=1+4+8, YSTYLE=1+4+8, XRANGE=[-40,40], YRANGE=[0,0.45], $
    THICK=3*(2*PASS+1)
;
; Make the axes and axes labels for the Gaussian in a non-standard way.
;
ARROW, -40, 0, 40, 0, /DATA, /SOLID
OPLOT, [-40, 39], [0, 0], THICK=3*(2*PASS+1)
FOR I = 1, 7 DO BEGIN
    XTPOS = I - 4
    CASE ABS(XTPOS) OF
        0:  STOUT = '0'
        1:  STOUT = SSIGMA(PASS)
     ELSE:  STOUT = STRTRIM(XTPOS, 2) + SSIGMA[PASS]
    ENDCASE
    OPLOT, [XTPOS, XTPOS]*SIGMA1, [0,0.02], THICK=3*(2*PASS+1)
    XYOUTS, XTPOS*SIGMA1, -0.03, '!6'+STOUT, SIZE=1.0, ALIGN=0.5, $
        FONT=AFONT[PASS]
ENDFOR
XYOUTS, 0, -0.06, '!6x - '+SMU[PASS], SIZE=1.2, ALIGN=0.5, FONT=AFONT[PASS]
ARROW, -40, 0, -40, 0.45, /DATA, /SOLID
OPLOT, [-40, -40], [0, 0.44], THICK=3*(2*PASS+1)
FOR I = 1, 4 DO BEGIN
    OPLOT, [0, 2]-4*SIGMA1, [I, I]*0.1, THICK=3*(2*PASS+1)
    XYOUTS, -4*SIGMA1-1, I*0.1-0.005, '!6'+STRING(FORMAT='(F3.1)',I*0.1), $
        SIZE=1.0, ALIGN=1.0, FONT=AFONT[PASS]
ENDFOR
XYOUTS, -48, 0.22, '!6P!DG!N(x,'+SMU[PASS]+'!6,'+SSIGMA[PASS]+'!6)', $
    SIZE=1.2, ALIGN=0.5, ORIENT=90, FONT=AFONT[PASS]
;
; Indicate the standard deviation marker.
```

```
;
OPLOT, [0, 0], YSIG1+[-0.01, 0.01], THICK=3*(2*PASS+1)
ARROW, -5., YSIG1, -0.5, YSIG1, /DATA, /SOLID
OPLOT, [-5, -0.55], [YSIG1, YSIG1], THICK=3*(2*PASS+1)
ARROW, SIGMA1+5., YSIG1, SIGMA1+0.5, YSIG1, /DATA, /SOLID
OPLOT, SIGMA1+[5, 0.55], [YSIG1, YSIG1], THICK=3*(2*PASS+1)
XYOUTS, SIGMA1+6., YSIG1-0.005, SSIGMA[PASS], SIZE=1.2, FONT=AFONT[PASS]
;
; Draw a circle on the Gaussian figure.
;
OPLOT, XCIR*5-24, YCIR*0.05+0.3, THICK=3*(2*PASS+1)
;
; This is the area where we generate a postscript file with pretty fonts.
; Here we adjust the postscript output so that the circle will appear
; "circular" in postscript.
;
IF PASS EQ 0 THEN BEGIN
    SET_PLOT, 'PS'
    DEVICE, XSIZE=8.0, XOFFSET = 0.25, YSIZE=5.0, YOFFSET=2.50, /INCHES, $
        /PORTRAIT, FILENAME='mycode2.ps'
    DEVICE, /SCHOOLBOOK, /BOLD, FONT_INDEX=17
    DEVICE, /SYMBOL, FONT_INDEX=7
    DEVICE, /BKMAN, /DEMI, FONT_INDEX=6
    DEVICE, /HELVETICA, /BOLD, FONT_INDEX=5
    PASS = 1
    GOTO, PLOTAGAIN
ENDIF ELSE BEGIN
    PASS = 0
    DEVICE, /CLOSE_FILE
    SET_PLOT, TERM
    PRINT, 'Plot saved in file:  mycode2.ps'
ENDELSE
;
RETURN
;
END
```

3 Compile and Run this Code.

To compile this code, click the “Compile” button icon near the top of the GUI just under the pull-down menu items. To run this procedure, click the “Run” arrow button next to the Compile button. The figure will appear on your screen and a postscript file named mycode2.ps will be generated in your current directory. Note that in order to view this postscript file on your terminal window, you will need to double-click on this file which will automatically run the GhostView software to view this file on the display. One should always review the postscript output before sending it to a printer.

When you are through working with this file, make sure you close it (in the "File" pull-down menu).

One can always access the IDL Help utility by typing a question mark (?) in the command line box near the bottom of the GUI or clicking on the Help tab in the top banner of the IDL GUI.

4 Finishing Up

Note that one can always check out what files are located on the current directory while in IDL by entering the following command at the IDL> prompt:

```
IDL> $ ls
```

where the '$' sign tells IDL that this is an operating system command and 'ls' is the Unix/Linux command for a directory listing. One can also find out the name of the current directory one is in by issuing the following at the IDL> prompt:

```
IDL> cd, current=mydir
IDL> help, mydir
```

where the 'current' keyword tells the 'cd' command to place the name of the current directory in the string variable 'mydir'. The help command will then give you the details of the 'mydir' variable. One could have also just printed the value stored in mydir with

```
IDL> print, mydir
```

We could have also retrieved the current directory from Unix/Linux with the following command in IDL:

```
IDL> $ pwd
```

where 'pwd' is the Unix command for 'print working directory.'

That's it for now. Make sure you exit out of IDL before logging off of your machine. One simply does this by either entering exit at the IDL prompt box near the bottom of the GUI or selecting the Exit item under the File pull-down menu.