1 Getting Started with IDL

IDL can be used as an interactive command language and as a programming language. The commands and programs run on a wide range of computers. The program files that are created on one computer can be run on another, usually without alteration.

IDL is used in many imaging applications, including astronomy, remote sensing and medicine. Large user communities are well established. These user groups maintain sites with libraries of specialized programs. The imaging, programming and platform independent capabilities of IDL are major reasons for adopting it as the foundation for this course.

This material is intended to provide a basic working knowledge of IDL. It certainly will not make you a proficient IDL programmer. However, it should provide you with the essential tools for this course.

I will assume that you are using IDL on a computer and an operating system that has a graphical user interface. I will also assume that you are using IDLDE, the IDL Development Environment.

The IDL Help system should be consulted frequently. IDL has a large number of commands and the commands have lots of options. You will probably find it impossible to remember all the details. If you don’t understand a command or one of its keywords (more about keywords soon) then be sure to go to the Help system and read about it. Along the way you will run across lots of other interesting commands and command options.

I will not try to duplicate or improve the descriptions in the IDL Help system. Rather, I will introduce a topic and then ask you to go to the Help system to get more information. This will be done in the context of examples. You will need to have IDLDE available as you read this material so that you can work interactively.

1.1 System Environment

IDL runs within the GUI environment on your computer. Your computer operating system manages the file system, keyboard, mouse, network interaction, and the displays (monitor, printer, plotter). When IDL makes a window and displays an image on a monitor it is actually doing that through calls to the operating system. Various events can be generated by the mouse and keyboard. This interaction is also handled through the operating system. Because IDL works in several operating systems, the details of these interactions and displays is bound to be somewhat different on the different systems. Among them is mouse interaction. Unix systems support three buttons, Windows systems support two buttons, and the Macintosh supports one button.

It is not really possible to address all of the possible variations within this material. I assume that you will be able to make suitable adjustments for your system by referring to the Help documentation and the documentation for your operating system.
If you are running IDL remotely using VNC, then you will need to have an account on the host system. The procedure for use of the VNC system is described on the course web page. You can invoke IDL on the remote system by entering `idlde` at the system prompt. The behavior should be essentially the same as what you would have on a local computer except for the effects of network delay.

1.2 IDL Development Environment

The first thing to do is to start the IDLDE on your system. This will give you a GUI window like that shown in Figure 1. This is just one possible configuration of the display. You will undoubtedly end up customizing your environment as time goes on.

![Graphical user interface for the IDL Development Environment](image)

Figure 1: Graphical user interface for the IDL Development Environment. Commands can be entered in the command line at the bottom. IDL provides output through the log window. You can write programs in the edit window. A menu bar and various useful icons are at the top.

You can interact with IDL by entering individual commands at the command
The first command you might want to try is Demo. It will bring up an IDL widget that will demonstrate some of IDL’s capabilities.

We will begin with using commands from the command line. We will then talk about building batch files, which are files that can be run from the command line to execute a command script. We will then talk about building IDL procedures and functions. Once you have a procedure or function you can execute it from the command line or include it in another procedure or function. We will solve the mystery about what is the difference between a procedure and a function. Along the way we will introduce some built-in commands, such as those to make windows, display images, find and open files, and print and plot.

1.3 Command Line Entry

Suppose that you want to have IDL print ‘Hello, World’ on the output window. You simply enter the following on the command line (type everything after the IDL> prompt and hit the enter key).

   IDL> Print,‘Hello, World’

You should see the string ‘Hello, World’ printed on the output (log) window.

   In this case the command is the word Print followed by a comma followed by a description of what to print. Simply put, we have given a command name and something for the command to process. In this case the input is a string to be printed. The command we used in this case is one of the many built-in procedures or functions of IDL.

   All IDL procedures (whether written by a user or built in) have the format.

   ProcName,parm1,parm2,...,parmn

   The procedure name represented by ProcName is followed by zero or more parameters. The parameters are constants or variables that serve to give information to the procedure (input parameters) or get information from it (output parameters). The only way to tell whether a particular parameter is input or output is to read the documentation for the procedure. A constant can be an input parameter but not an output parameter. (Why?)

   The following procedure creates a blank graphics window that can be used to display a plot or an image.

   IDL> Window,3,Xsize=400,Ysize=300,Title=‘My Window’

   The result will be a window that is 400 pixels wide by 300 pixels high and has the title ‘My Window’. The window index is 3. The index is used to select among the open windows if there is more than one. In this example, the procedure name is Window, and there are four parameters. The first parameter, the index, is a positional parameter and the other three are keyword parameters.

   A positional parameter is identified by its location and a keyword parameter is identified by its name. The format for a keyword parameter is Name=Value. You will learn that you can to choose whether to make a parameter positional or keyword when you write a program. There is no hard and fast rule about which
way to go. Once you have made the choice, anyone who uses your program has
to follow that format.

Let’s create something to plot in our graphics window. Along the way we
will introduce a couple of built-in functions. Enter the following:

IDL> t=FindGen(101)
IDL> y=t*Sin(2*!Pi*(t/50)^2)
IDL> Plot,t,y,Title='Modulated Sinusoid'

This graph is shown in Figure 2. The graph is generated by the Plot procedure.
We have given it three parameters: the data vectors t and y and the title of the
graph. t and y are positional parameters and Title is a keyword parameter. The
positional parameters must be in the order expected by the procedure. You can
check this by entering the command

Plot,y,t,Title='A New Plot'.

Figure 2: Modulated sinusoidal function plotted in a window.

The vectors t and y use some IDL functions. The command

\[ t = \text{FindGen}(101) \]

creates the vector \( t = [0.0, 1.0, 2.0, \ldots, 100.0] \) that contains 101 elements. The
entry in position \( n \) is \( t_n = n \). All entries are floating point numbers. A function
command is of the form
value=FunctionName(parm1,parm2,...parmn)

The parameters can be either positional or keyword constants or variables. (Again, the output parameters cannot be constants) There may be zero or more parameters contained in the parentheses, and they may serve as inputs or as outputs. Functions always return a value to the parameter on the left of the equal sign.

Technically, the variable $t$ is an array with one row and 101 columns. We can print out the first few numbers in $t$ with the following

IDL> Print,'t=',t[0:5]

The result is

$$t = 0.000000 \ 1.00000 \ 2.00000 \ 3.00000 \ 4.00000 \ 5.00000$$

In this case we have used \texttt{Print} with two parameters, a string and a subscripted array. It handles both in a sensible manner. We will discuss selecting ranges of subscripts later, but this is an example.

The variable $y$ is also an array that is constructed by passing $t$ into the sine function. All of the values in $t$ are processed in one statement. The array $y$ is the same size as $t$.

The symbol $\texttt{!Pi}$ is the IDL way to represent the number $\pi$. It is one of many built-in system constants. All system constants and variables start with an exclamation mark.

IDL is not case sensitive. Except within quote marks, upper case and lower case letters mean exactly the same thing. This has good and bad consequences. A bad consequence is that the variables $A$ and $a$ mean the same thing, so you can’t have an upper case and lower case symbol for different things. I would like to let $A$ stand for the Fourier transform of $a$ in certain programs. But, I can’t do that. On the other hand, if I mistakenly use the wrong case nothing is harmed.

The \texttt{Plot} procedure graphs $y$ vs $t$. When \texttt{Plot} is given two positional parameters it treats the first as the ordinate and the second as the abscissa. \texttt{Plot} also has many keyword parameters, among which is Title. You should read about \texttt{Plot} in the Help system and perhaps try out a few of its keyword options.

By now you will have observed an interesting thing about IDL procedures and functions. The number of parameters used in a call of a command is not fixed. The \texttt{Print} command has been used with different arguments in different arrangements. \texttt{Plot} is another procedure with this quality. Try

IDL> Plot,y

Interestingly, you get essentially the same graph as with the earlier plot command. When \texttt{Plot} is used with just one positional argument it uses it for the vertical values and supplies an automatic value that starts at zero and increases by one unit per value for the horizontal axis. You only have to supply the vector for the ordinates in this quick-and-dirty plot command.
This has been a quick introduction to some of the basic interactive elements of IDL. You should now be able to read along and duplicate the commands in the text.