

The Command Window

Overview	20
Command Window Example.....	20
The Command Buffer	21
Command Window Title	22
History Area	22
Limiting Command History.....	22
History Archive.....	22
History Carbon Copy	22
Searching the Command Window	24
Command Window Formats.....	24
Getting Help from the Command Line	24
Command Window Shortcuts	25

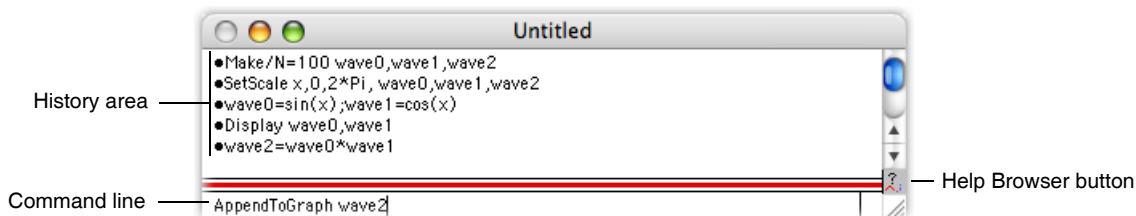
Overview

You can control Igor using menus and dialogs or using commands that you execute from the command window. Some actions, for example waveform assignments, are much easier to perform by entering a command. Commands are also convenient for trying variations on a theme — you can modify and reexecute a command very quickly. If you use Igor regularly, you may find yourself using commands more and more for those operations that you frequently perform.

In addition to executing commands in the Command window, you can also execute commands in a notebook, procedure or help window. These techniques are less commonly used than the command window. See **Notebooks as Worksheets** on page III-5 for more information.

This chapter describes the command window and general techniques and shortcuts. See Chapter IV-1, **Working with Commands**, for details on command usage and syntax.

The command window consists of a **command line** and a **history area**. When you enter commands in the command line and press Return (*Macintosh*) or Enter (*Windows and Macintosh*), the commands are executed. Then they are saved in the history area for you to review. If a command produces text output, that output is also saved in the history area. A bullet character is prepended to command lines in the history so that you can easily distinguish command lines from output lines.



The Command window includes a help button just below the History area scroll bar. Clicking the button displays the Help Browser window. See **Igor Help Browser** on page II-6 for more details about the Help Browser.

The total length of a command on the command line must not exceed 400 characters.

There is no line continuation character in Igor. However, it is nearly always possible to break a single command up into multiple lines.

Command Window Example

Here is a quick example designed to illustrate the power of commands and some of the shortcuts that make working with commands easy.

1. **Choose New Experiment from the File menu.**
2. **Execute the following command by typing in the command line and then pressing Return or Enter.**
Make/N=100 wave0; Display wave0
This displays a graph.
3. **Press Command-J (Macintosh) or Ctrl+J (Windows).**
This activates the command window.
4. **Execute**
SetScale x, 0, 2*PI, wave0; wave0 = sin(x)
The graph shows the sine of x from 0 to 2π .
Now we are going to see how to quickly retrieve, modify and reexecute a command.
5. **Press the Up Arrow key.**
This selects the command that we just executed.
6. **Press Return or Enter.**
This transfers the selection back into the command line.

7. Change the “2” to “4”.

The command line should now contain:

```
SetScale x, 0, 4*PI, wave0; wave0 = sin(x)
```

8. Press Return or Enter to execute the modified command.

This shows the sine of x from 0 to 4π .

9. While pressing Option (Macintosh) or Alt (Windows), click the last command in the history.

This is another way to transfer a command from the history to the command line. The command line should now contain:

```
SetScale x, 0, 4*PI, wave0; wave0 = sin(x)
```

10. Press Command-K (Macintosh) or Ctrl+K (Windows).

This “kills” the contents of the command line.

Now let’s see how you can quickly reexecute a previously executed command.

11. With Command and Option (Macintosh) or Ctrl and Alt (Windows) pressed, click the second-to-last command in the history.

This reexecutes the clicked command (the 2π command).

Repeat this step a number of times, clicking the second-to-last command each time. This will alternate between the 2π command and the 4π command.

12. Execute

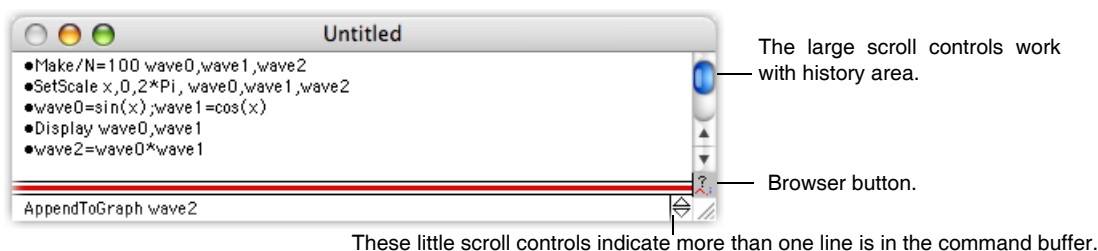
```
WaveStats wave0
```

Note that the WaveStats operation has printed its results in the history where you can review them. You can also copy a number from the history to paste into a notebook or an annotation.

There is a summary of all command window shortcuts at the end of this chapter.

The Command Buffer

The command line shows a single line of the **command buffer**. Normally the command buffer is either empty or contains just one line of text. However you can copy multiple lines of text from any window and paste them in the command buffer. When more than one line is in the command buffer, little scroll controls appear at the right end of the command line.



You can clear the contents of the command buffer by choosing the Clear Command Buffer item in the Edit menu or by pressing Command-K (Macintosh) or Ctrl+K (Windows).

When you invoke an operation from a typical Igor dialog, the dialog puts a command in the command buffer and executes it. The command is then transferred to the history as if you had entered the command manually.

If an error occurs during the execution of a command, Igor leaves it in the command buffer so you can edit and reexecute it. If you don’t want to fix the command, you should remove it from the command buffer by pressing Command-K (Macintosh) or Ctrl+K (Windows).

Because the command buffer usually contains nothing or one command, we usually think of it as a single line and use the term “command line”.

Command Window Title

The title of the command window is the name of the experiment that is currently loaded. When you first start Igor or if you choose New from the File menu, the title of the experiment and therefore of the command window is “Untitled”.

When you save the experiment to a file, Igor sets the name of the experiment to the file name minus the file extension. If the file name is “An Experiment.pxp”, the experiment name is “An Experiment”. Igor displays “An Experiment” as the command window title.

For use in procedures, the **IgorInfo(1)** function returns the name of the current experiment.

History Area

The history area is a repository for commands and results.

Text in the history area can not be edited but can be copied to the Clipboard or to the command line. Copying text to the Clipboard is done in the normal manner. To copy a command from the history to the command buffer, select the command in the history and press Return or Enter. An alternate method is to press Option (*Macintosh*) or Alt (*Windows*) and click in the history area.

To make it easy to copy a command from the history to the command line, clicking a line in the history area selects the entire line. You can still select just part of a line by clicking and dragging.

Up Arrow and Down Arrow move the selection range in the history up or down one line selecting an entire line at a time. Since you normally want to select a line in the history to copy a command to the command line, Up Arrow and Down Arrow skip over non-command lines. Left Arrow and Right Arrow move the insertion point in the command line.

When you save an experiment, the contents of the history area are saved. The next time you load the experiment the history will be intact. Some people have the impression that Igor recreates an experiment by reexecuting the history. This is not correct. See **How Experiments Are Loaded** on page II-39 for details.

Limiting Command History

The contents of the history area can grow to be quite large over time. You can limit the number of lines of text retained in the history using the Limit Command History feature in the Command Settings section of the Miscellaneous Settings dialog which is accessible through the Misc menu.

If you limit command history, when you save the experiment, Igor checks the number of history lines. If they exceed the limit, the oldest lines are deleted.

History Archive

When history lines are deleted through the Limit Command History feature, the History Archive feature allows you to tell Igor to write the deleted lines to a text file in the experiment's home folder.

To enable the History Archive feature for a given experiment, create a plain text file in the home folder of the experiment. The text file must be named "<Experiment Name> History Archive.txt" where <Experiment Name> is the name of the current experiment. Now, when you save the experiment, Igor writes any deleted history lines to the history archive file.

If the history archive file is open in any program, including Igor, the history archive feature will fail and no history lines will be written.

History Carbon Copy

This feature is expected to be of interest only in rare cases for advanced Igor programmers such as Bela Farago who requested it.

You can designate a notebook to be a "carbon copy" of the history area by creating a plain text or formatted notebook and setting its window name, via Windows->Window Control, to HistoryCarbonCopy. If the HistoryCarbonCopy notebook exists, Igor inserts history text in the notebook as well as in the history. However, if a command is initiated from the HistoryCarbonCopy notebook (see **Notebooks as Worksheets** on page III-5), Igor suspends sending history text to that notebook during the execution of the command.

If you rename the notebook to something other than HistoryCarbonCopy, Igor will cease sending history text to it. If you later rename it back to HistoryCarbonCopy, Igor will resume sending history text to it.

The history trimming feature accessed via the Miscellaneous Settings dialog does not apply to the HistoryCarbonCopy notebook. You must trim it yourself. Notebooks are limited to 16 million paragraphs.

When using a formatted notebook as the history carbon copy, you can control the formatting of commands and results by creating notebook rulers named Command and Result. When Igor sends text to the history carbon copy notebook, it always applies the Command ruler to commands. It applies the Result ruler to results if the current ruler is Normal, Command or Result. You must create the Command and Result rulers if you want Igor to use them when sending text to the history carbon copy.

This function creates a formatted history carbon copy notebook with the Command and Result rulers used automatically by Igor as well as an Error ruler which we will use for our custom error messages:

```
Function CreateHistoryCarbonCopy()
    NewNotebook /F=1 /N=HistoryCarbonCopy /W=(50,50,715,590)

    Notebook HistoryCarbonCopy backRGB=(0,0,0)// Set background to black

    Notebook HistoryCarbonCopy showRuler=0

    // Define ruler to govern commands.
    // Igor will automatically apply this to commands sent to history carbon copy.
    Notebook HistoryCarbonCopy newRuler=Command,
        rulerDefaults={"Geneva",10,0,(65535,65535,0)}

    // Define ruler to govern results.
    // Igor will automatically apply this to results sent to history carbon copy.
    Notebook HistoryCarbonCopy newRuler=Result,
        rulerDefaults={"Geneva",10,0,(0,65535,0)}

    // Define ruler to govern user-generated error messages.
    // We will apply this ruler to error messages that we send
    // to history carbon copy via Print commands.
    Notebook HistoryCarbonCopy newRuler=Error,
    rulerDefaults={"Geneva",10,0,(65535,0,0)}
End
```

If the current ruler is not Normal, Command or Result, it is assumed to be a custom ruler that you want to use for special messages sent to the history using the Print operation. In this case, Igor does not apply the Result ruler but rather allows your custom ruler to remain in effect.

This function sends an error message to the history using the custom Error ruler in the history carbon copy notebook:

```
Function PrintErrorMessage(message)
    String message

    Notebook HistoryCarbonCopy, ruler=Error
    Print message

    // Set ruler back to Result so that Igor's automatic use of the Command
    // and Result rulers will take effect for subsequent commands.
```

Chapter II-2 — The Command Window

```
Notebook HistoryCarbonCopy, ruler=Result
End
```

XOP programmers can use the XOPNotice3 XOPSupport routine to control the color of text sent to the History Carbon Copy notebook.

Searching the Command Window

You can search the command line or the history by choosing Find from the Edit menu or by using the keyboard shortcuts as shown in the Edit menu. Searching the command line is most often used to modify a previously executed command before reexecuting it. For example, you might want to replace each instance of a particular wave name with another wave name.

If there is an active selection in the history, Find searches the history. Otherwise it searches the command line. To be sure that Find will search the area that you want, you can click in that area before starting the search.

Command Window Formats

You can change the text format used for the command line. For example, you might prefer larger type. To do this, click in the command line and then choose Set Text Format from the Command Buffer submenu of the Misc menu. To set the text format for the history area, click in the history area and then choose Set Text Format from the History Area submenu of the Misc menu. To do this, the history area must have some text in it.

You can set other properties, such as background color, by choosing Document Settings from the Command Buffer or History Area submenus. The Document Settings dialog also sets the header and footer used when printing the history.

When you change the text format or document settings, you are changing the current experiment only. You may want to capture the new format and settings as a preference for new experiments. To do this, choose Capture Prefs from the Command Buffer and History Area submenus.

Getting Help from the Command Line

When working with the command line, you might need help in formulating a command. There are shortcuts that allow you to insert a template, view help, or find the definition of a user function.

To insert a template, type the name of the operation or function and then press Shift-Help (*Macintosh*) or Ctrl+F1 (*Windows*).

To view help or to view the definition of a user function, type the name of the operation or function and then press Shift-Option-Help (*Macintosh*) or Ctrl+Alt+F1 (*Windows*).

You can also insert a template or get help by Control-clicking (*Macintosh*) or right-clicking (*Windows*).

To view text window keyboard navigation shortcuts, see **Text Window Navigation** on page II-68.

This table may help you remember what the various keyboard shortcuts do.

Keyboard Shortcut		What It Does
<i>Macintosh</i>	<i>Windows</i>	
Press Help	Press F1	Displays help browser window
Press Shift-Help	Press Ctrl+F1	Inserts template for selected operation or function
Press Shift-Option-Help	Press Ctrl+Alt+F1	Displays help for selected operation or function

Command Window Shortcuts

Action	Shortcut (<i>Macintosh</i>)	Shortcut (<i>Windows</i>)
To activate the command window	Press Command-J.	Press Ctrl+J.
To clear the command buffer	Press Command-K.	Press Ctrl+K.
To get a contextual menu of commonly-used actions	Press Control and click the history area or command line	Right-click the history area or command line
To copy a line from the history to the command buffer	Click the line and press Return or Enter. Press Option and click the line.	Click the line and press Enter. Press Alt and click the line.
To reexecute a line from the history	Click the line and press Return or enter twice Press Command-Option and click the line.	Click the line and press Ctrl+Enter Press Ctrl+Alt and click the line.
To find a recently executed command in the history	Press the Up or Down Arrow keys.	Press the Up or Down Arrow keys.
To find text in the history	Click in the history area and press Command-F.	Click in the history area and press Ctrl+F.
To find text in the command line	Click in the command line and press Command-F.	Click in the command line and press Ctrl+F.
To get a template	Type the name of an operation or function and press Shift-Help.	Type the name of an operation or function and press Ctrl+F1.
To get help or view the definition of a user function	Type the name of an operation or function and press Shift-Option-Help.	Type the name of an operation or function and press Ctrl+Alt+F1.
