Chapter
I-1

Introduction to Igor Pro

Introduction to Igor Pro

Igor Pro is an integrated program for visualizing, analyzing, transforming and presenting experimental data.

Igor Pro's features include:

- Publication-quality graphics
- High-speed data display
- Ability to handle large data sets
- Curve-fitting, Fourier transforms, smoothing, statistics, and other data analysis
- Waveform arithmetic
- Image display and processing
- Combination graphical and command-line user interface
- Automation and data processing via a built-in programming environment
- Extensibility through modules written in the C and C++ languages

Some people use Igor simply to produce high-quality, finely-tuned scientific graphics. Others use Igor as an all-purpose workhorse to acquire, analyze and present experimental data using its built-in programming environment. We have tried to write the Igor program and this manual to fulfill the needs of the entire range of Igor users.

The Igor application is shipped in both 32-bit and 64-bit versions. We refer to the 64-bit version as "Igor64". You should use the 64-bit version except for rare cases where compatibility with 32-bit plug-ins is required.

Igor Objects

The basic objects that all Igor users work with are:

- Waves
- Graphs
- Tables
- Page layouts

A collection of objects is called an "experiment" and is stored in an experiment file. When you open an experiment, Igor recreates the objects that comprise it.

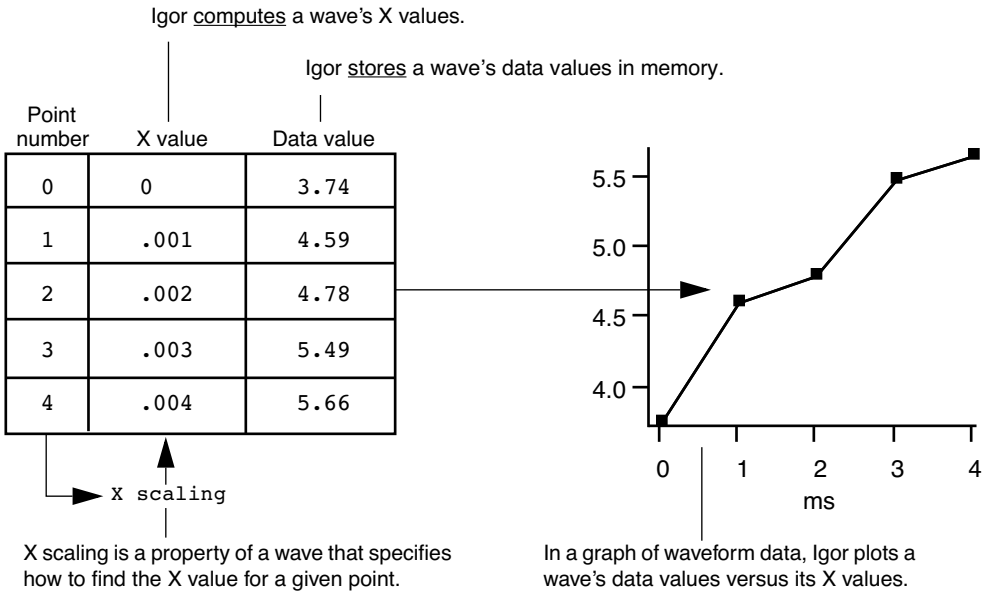
Waves — The Key Igor Concept

We use the term "wave" to describe the Igor object that contains an array of numbers. Wave is short for "waveform". The wave is the most important Igor concept.

Igor was originally designed to deal with waveform data. A waveform typically consists of hundreds to thousands of values measured at evenly-spaced intervals of time. Such data is usually acquired from a digital oscilloscope, scientific instrument or analog-to-digital converter card.

The distinguishing trait of a waveform is the *uniform spacing* of its values along an axis of time or other quantity. An Igor wave has an important property called "X scaling" that you set to specify the spacing of your data. Igor *stores* the Y component for each point of a wave in memory but it *computes* the X component based on the wave's X scaling.

In the following illustration, the wave consists of five data points numbered 0 through 4. The user has set the wave's X scaling such that its X values start at 0 and increment by 0.001 seconds per point. The graph displays the wave's stored data values versus its computed X values.



Waves can have from one to four dimensions and can contain either numeric or text data.

Igor is also capable of dealing with data that does not fit the waveform metaphor. We call this XY data. Igor can treat two waves as an XY pair. In an XY pair, the data values of one wave supply the X component and the data values of another wave supply the Y component for each point in the pair.

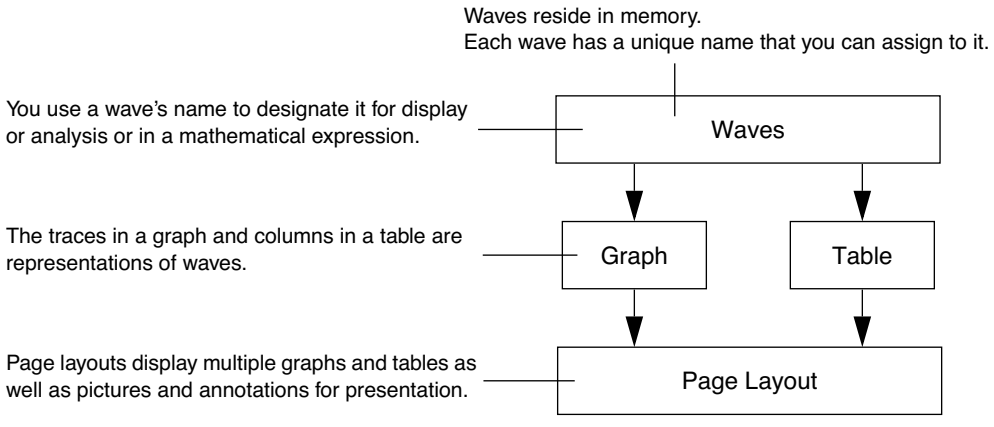
A few analysis operations, such as Fourier transforms, inherently work only on waveform data. They take a wave's X scaling into account.

Other operations work equally well on waveform or XY data. Igor can graph either type of data and its powerful curve fitting works on either type.

Most users create waves by loading data from a file. You can also create waves by typing in a table, evaluating a mathematical expression, acquiring from a data acquisition device, and accessing a database.

How Objects Relate

This illustration shows the relationships among Igor's basic objects. Waves are displayed in graphs and tables. Graphs and tables are displayed in page layouts. Although you can display a wave in a graph or table, a wave does not need to be displayed to exist.



Each object has a name so that it can be referenced in an Igor command. You can explicitly set an object's name or accept a default name created by Igor.

Graphs are used to visualize waves and to generate high-quality printouts for presentation. The traces in a graph are representations of waves. If you modify a wave, Igor automatically updates graphs. Igor labels the axes of a graph intelligently. Tick marks never run into one another and are always “nice” values no matter how you zoom in or pan around.

In addition to traces representing waveform or XY data, a graph can display an image or a contour plot generated from 2D data.

Tables are used to enter, inspect or modify wave data. A table in Igor is not the same as a spreadsheet in other graphing programs. A column in a table is a *representation* of the contents of a wave. The wave continues to exist even if you remove it from the table or close the table entirely.

Page layouts permit you to arrange multiple graphs and tables as well as pictures and annotations for presentation. If you modify a graph or table, either directly or indirectly by changing the contents of a wave, Igor automatically updates its representation in a layout.

Both graphs and layouts include drawing tools for adding lines, arrows, boxes, polygons and pictures to your presentations.

More Objects

Here are some additional objects that you may encounter:

- Numeric and string variables
- Data folders
- Notebooks
- Control panels
- 3D plots
- Procedures

A numeric variable stores a single number and a string variable stores a text string. Numeric and string variables are used for storing bits of data for Igor procedures.

A data folder can contain waves, numeric variables, string variables and other data folders. Data folders provide a way to keep a set of related data, such as all of the waves from a particular run of an experiment, together and separate from like-named data from other sets.

A notebook is like a text-editor or word-processor document. You can use a notebook to keep a log of results or to produce a report. Notebooks are also handy for viewing Igor technical notes or other text documentation.

A control panel is a window containing buttons, checkboxes and other controls and readouts. A control panel is created by an Igor user to provide a user interface for a set of procedures.

A 3D plot displays three–dimensional data as a surface, a scatter plot, or a path in space.

A procedure is a programmed routine that performs a task by calling Igor's built-in operations and functions and other procedures. Procedures range from very simple to very complex and powerful. You can run procedures written by WaveMetrics or by other Igor users. If you are a programmer or want to learn programming, you can learn to write your own Igor procedures to automate your work.

Igor’s Toolbox

Igor's toolbox includes a wide range of built-in routines. You can extend it with user-defined procedures written in Igor itself and with separately-compiled Igor extensions (plug-ins) that you obtain from WaveMetrics, from a colleague, from a third-party, or write yourself.

Built-In Routines

Each of Igor's built-in routines is categorized as a function or as an operation.

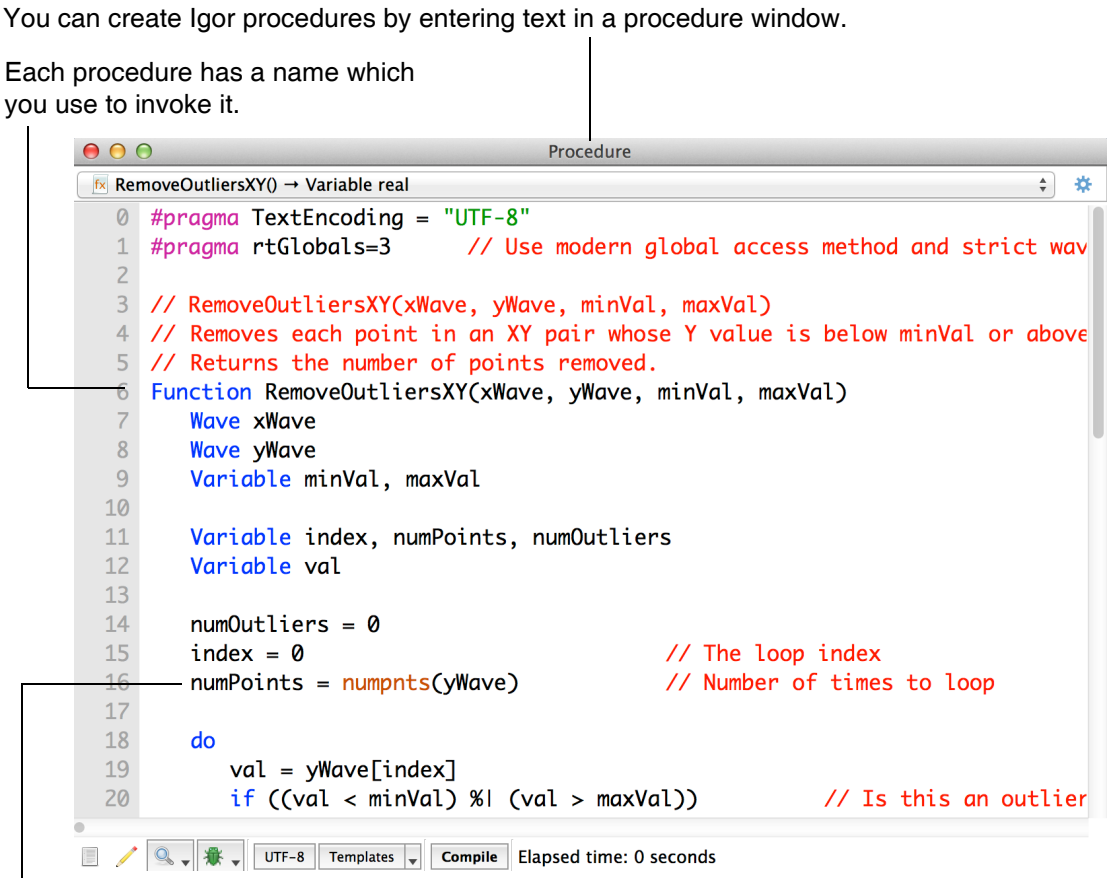
A built-in function is an Igor routine, such as sin, exp or ln, that directly returns a result. A built-in operation is a routine, such as Display, FFT or Integrate, that acts on an object and may create new objects but does not directly return a result.

A good way to get a sense of the scope of Igor's built-in routines is to scan the sections **Built-In Operations by Category** on page V-1 and **Built-In Functions by Category** on page V-7 in the reference volume of this manual.

For getting reference information on a particular routine it is usually most convenient to choose Help→Command Help and use the Igor Help Browser.

User-Defined Procedures

A user-defined procedure is a routine written in Igor's built-in programming language by entering text in a procedure window. It can call upon built-in or external functions and operations as well as other user-defined procedures to manipulate Igor objects. Sets of procedures are stored in procedure files.



Igor Extensions

An extension is a "plug-in" - a piece of external C or C++ code that adds functionality to Igor. For historical reasons, we use the term "XOP" to refer to an Igor extension. "XOP" is a contraction of "external operation". The terms "XOP" and "Igor extension" are synonymous.

Originally XOPs were intended to allow adding operations only to Igor. Now XOPs can add much more, including functions, menus, dialogs, and windows, so "XOP" has the meaning "external module that extends Igor".

To create an XOP, you must be a C or C++ programmer and you need the optional **Igor External Operations Toolkit**. See **Creating Igor Extensions** on page IV-195.

Although *creating* an extension is a job for a programmer, anyone can *use* an extension. The Igor installer automatically installs commonly used extensions in "Igor Pro 7 Folder/Igor Extensions (64-bit)". These extensions are available for immediate use.

Less commonly used extensions are installed in "Igor Pro 7 Folder/More Extensions (64-bit)". Available extensions are described in the "XOP Index" help file (choose Help→Help Windows→XOP Index.ihf). To activate an extension, see **Activating Extensions** on page III-450.

Igor's User Interface

Igor uses a combination of the familiar graphical user interface and a command-line interface. This approach gives Igor both ease-of-use and programmability.

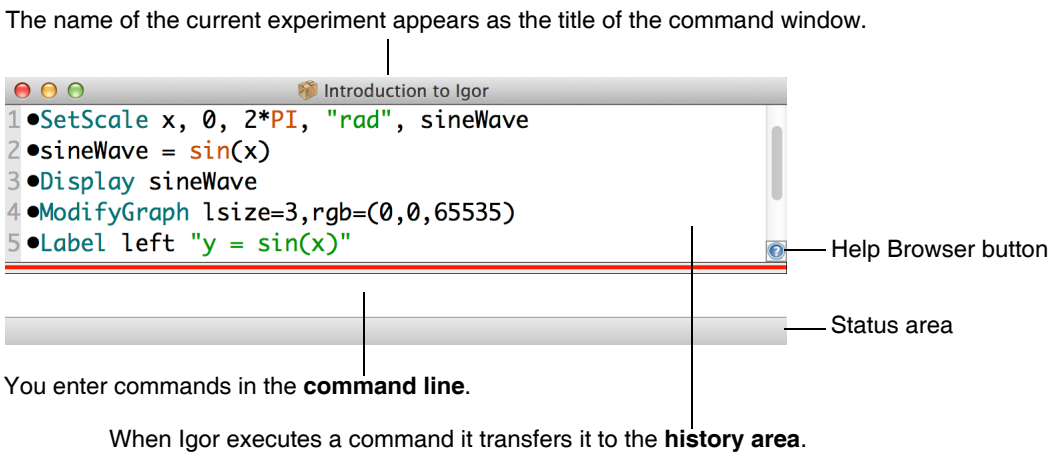
The job of the user interface is to allow you to apply Igor's operations and functions to objects that you create. You can do this in three ways:

- Via menus and dialogs
- By typing Igor commands directly into the command line
- By writing Igor procedures

The Command Window

The command window is Igor's control center. It appears at the bottom of the screen.

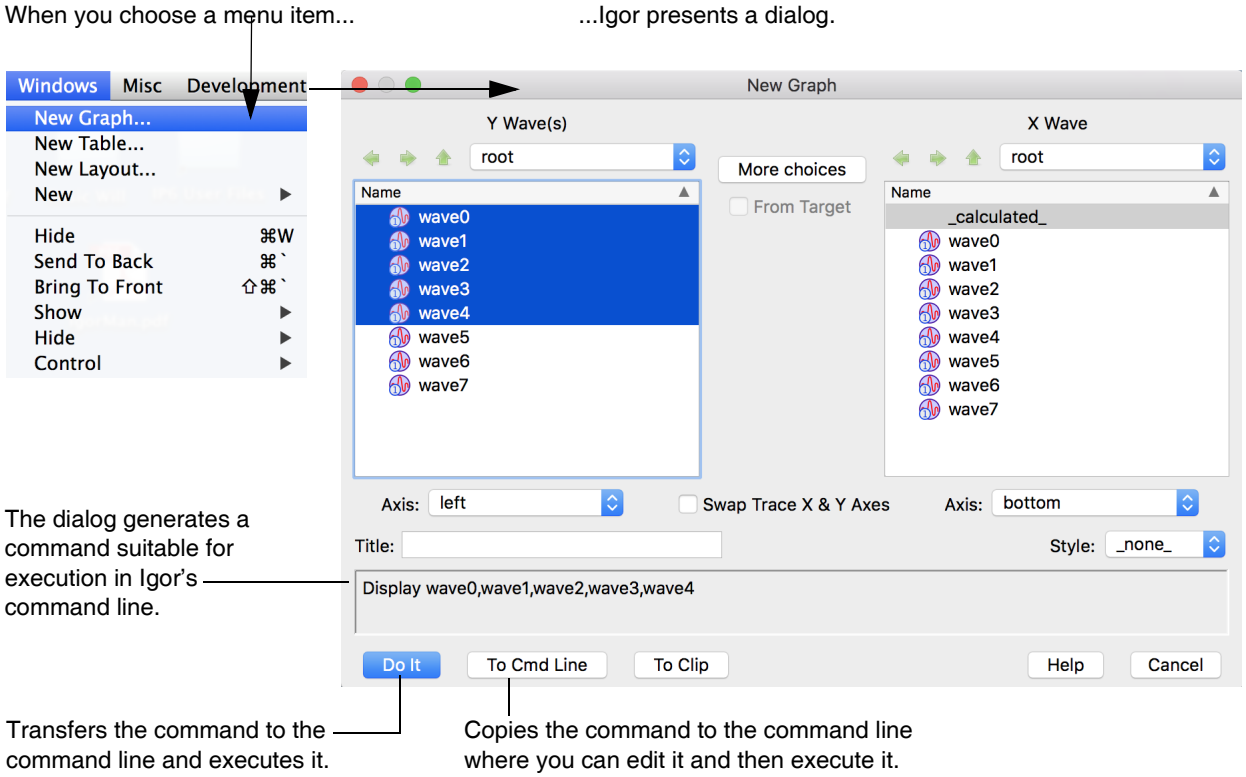
At the bottom of the command window is the command line. Above the red divider is the history area where executed commands are stored for review. Igor also uses the history area to report results of analyses like curve-fitting or waveform statistics.



Menus, Dialogs and Commands

Menus and dialogs provide easy access to the most commonly-used Igor operations.

When you choose a menu item associated with an Igor operation, Igor presents a dialog. As you use the dialog, Igor generates a command and displays it in the **command box** near the bottom of the dialog. When you click the Do It button, Igor transfers the command to the command line where it is executed.



As you get to know Igor, you will find that some commands are easier to invoke from a dialog and others are easier to enter directly in the command line.

There are some menus and dialogs that bypass the command line. Examples are the Save Experiment and Open Experiment items in the File menu.

Using Igor for Heavy-Duty Jobs

If you generate a lot of raw data or need to do custom technical computing, you will find it worthwhile to learn how to put Igor to heavy-duty use. It is possible to automate some or all of the steps involved in loading, processing, and presenting your data. To do this, you must learn how to write Igor procedures.

Igor includes a built-in programming environment that lets you manipulate waves, graphs, tables and all other Igor objects. You can invoke built-in operations and functions from your own procedures to build higher-level operations customized for your work.

Learning to write Igor procedures is easier than learning a lower-level language such as FORTRAN or C. The Igor programming environment is interactive so you can write and test small routines and then put them together piece-by-piece. You can deal with very high level objects such as waves and graphs but you also have fine control over your data. Nonetheless, it is still programming. To master it requires an effort on your part.

The Igor programming environment is described in detail in **Volume IV Programming**. You can get started by reading the first three chapters of that volume.

You can also learn about Igor programming by examining the WaveMetrics Procedures and example experiments that were installed on your hard disk.

Igor Documentation

Igor includes an extensive online help system and a comprehensive PDF manual.

The online help provides guided tours, tool tips, general usage information for all aspects of Igor, and reference information for Igor operations, functions and keywords.

The PDF manual contains the same information except for the tool tips.

The PDF manual, being in book format, is better organized for linear reading while the online help is usually preferred for reference information.

Tool Tips

Igor displays tool tips when you hover the mouse over an icon or dialog item.

You can turn tool tips off using the Miscellaneous Settings dialog. Choose Misc→Miscellaneous Settings, click the Help icon on the left, and uncheck the Show Tool Tips checkbox.

The Igor Help System

The Help menu provides access to Igor's help system, primarily through the Igor Help Browser.

- To display the Igor Help Browser, use the Help menu, click the question-mark icon in the command window, or press F1 (*Windows only*).
- Use the Igor Help Browser Help Topics tab to browse help topics.
- Use the Igor Help Browser Shortcuts tab to get a list of handy shortcuts and techniques.
- Use the Igor Help Browser Command Help tab to get reference information on Igor operations and functions. You can also right-click operation and function names in Igor windows to access the reference help.
- Use the Igor Help Browser Search tab to search Igor help, procedure and example files.

Most of the information displayed by the help browser comes from help files that are automatically loaded at launch time. The Help→Help Windows submenu provides direct access to these help files.

The Igor Manual

The Igor PDF manual resides in "Igor Pro 7 Folder/Manual". You can access it by choosing Help→Manual.

The manual consists of five volumes and an index.

Volume I contains the Getting Started material, including the Guided Tour of Igor Pro.

Volumes II and III contain general background and usage information for all aspects of Igor other than programming.

Volume IV contains information for people learning to do Igor programming.

Volume V contains reference information for Igor operations, functions and keywords.

Hard copy of the manual is not available.

Learning Igor

To harness the power of Igor, you need to understand its fundamental concepts. Some of these concepts are familiar to you. However, Igor also incorporates a few concepts that will be new to you and may seem strange at first. The most important of these are *waves* and *experiments.*

In addition to this introduction, the primary resources for learning Igor are:

- The **Guided Tour of Igor Pro** in Chapter I-2

  The guided tour shows you how to perform basic Igor tasks step-by-step and reinforces your understanding of basic Igor concepts along the way.

  The guided tour provides an essential orientation to Igor and is highly-recommended for all Igor users.

- The Igor Pro PDF manual and online help files

  You can access the PDF manual through Igor's Help menu or by opening it directly from the Manual folder of the Igor Pro 7 Folder where Igor is installed.

  You can access the help files through the Igor Help Browser (choose Help→Igor Help Browser) or directly through the Help→Help Windows submenu.

- The example experiments

  The example experiments illustrate a wide range of Igor features. They are stored in the Examples folder in the Igor Pro 7 Folder. You can access them using the File→Example Experiments submenu or directly from the Examples folder of the Igor Pro 7 Folder where Igor is installed.

You will best learn Igor through a combination of doing the guided tour, reading select parts of the manual (see suggestions following the guided tour), and working with your own data.

Videos of the guided tour are available at:

https://www.wavemetrics.com/igorpro/videotutorials.htm

Getting Hands-On Experience

This introduction has presented an overview of Igor, its main constituent parts, and its basic concepts. The next step is to get some hands-on experience and reinforce what you have learned by doing the **Guided Tour of Igor Pro** on page I-11.