Introduction to Data Analysis with IGOR Pro

IGOR Pro

WaveMetrics Igor Pro (9.1)
Based on slides by Adi Robinson

Introduction to Igor Pro

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

IGOR Pro 9.xx can be downloaded from the TWiki
To activate IGOR use the following information:
Serial Number: 933093
Activation Key: KUDY-ZNMK-YKXA-JXSP-ZEZM-ASJV-MJL

Igor Pro's features include:

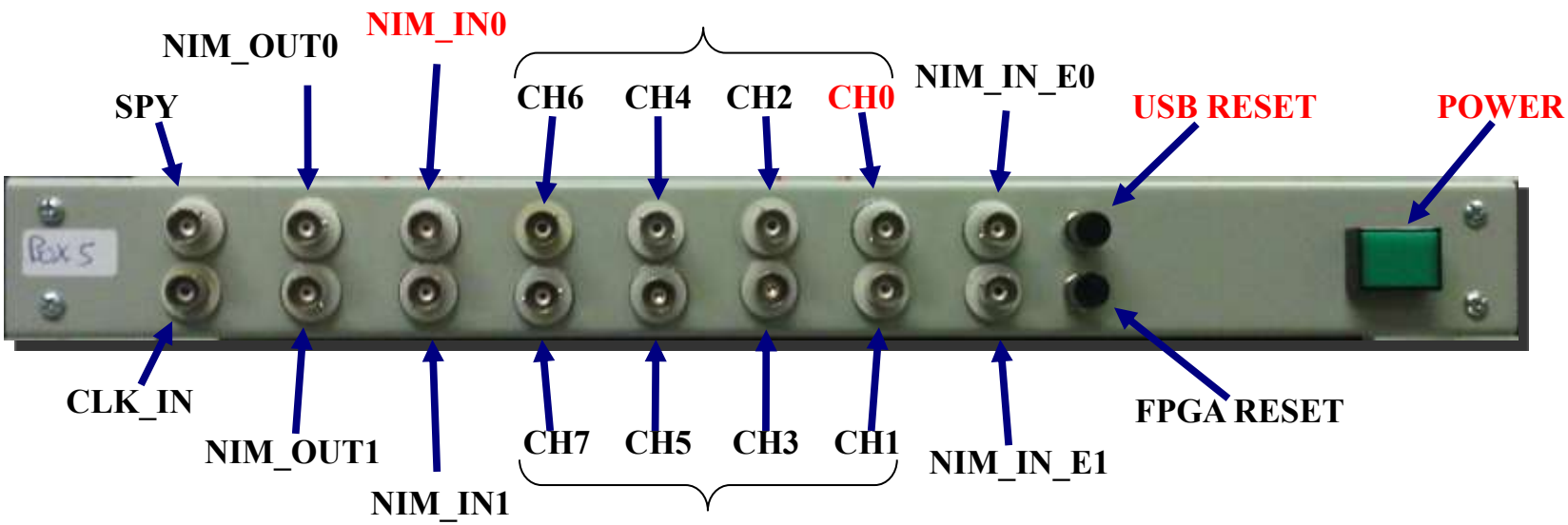
- Publication-quality graphics
- High-speed data display
- Ability to handle large event-by-event data sets
- Curve-fitting, Fourier transforms, smoothing, statistics, and other data analysis algorithms
- Waveform arithmetic
- Image display and processing
- Combination graphical and command-line user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

ANSEL Digital Data Source DDC-8 DSP

Detectors & electronics produce signal amplitude (energy) and time signals in up to 6 dimensions (≤ 6 signals per event).

Analog inputs

+/- 1V (maximum amplitude = 2V !!)



Analog inputs

+/- 1V (maximum amplitude = 2V !!)

Igor Pro - Waves

We use the term "wave" to describe an Igor object that contains an array of numbers. "Wave" is short for "waveform", a term used in digital signal processing (DSP). The *wave* is the most important Igor concept.

In the context of ANSEL experiments, a typical wave consists of a sequence of numbers describing a series of "events," for example, the stream of signal amplitudes generated by an operating radiation detector during some period of time.

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 are usually acquired from a digital oscilloscope, from a scientific instrument or from an analog-to-digital converter.

Igor Pro – Loading Waves

Most Igor users create “waves” by loading data from a file created by another program. In ANSEL, these original files are produced by the DDC8-Data Acquisition routines.

The process of *loading a file* reserves an array of computer cells and then stores data from the file in these cells. The waves can contain numeric or text data.

Optionally, one can create a new wave or overwrite an already existing wave.

Igor provides routines for loading files with a number of different data types.

(There is no single universal file format for numeric or text data that all programs can read and write.)

Igor Pro – How to start

IGOR Pro Manual

Guided Tour 1 - General Tour

In this exercise, we will generate data in three ways (typing, loading, and synthesizing) and we will generate graph, table, and page layout windows. We will jazz up a graph and a page layout with a little drawing and some text annotation. At the end we will explore some of the more advanced features of Igor Pro.

Launching Igor Pro

The Igor Pro application is typically installed in:

```
/Applications/Igor Pro Folder (Macintosh)
C:\Program Files\WaveMetrics\Igor Pro Folder (Windows 32-bit)
C:\Program Files (x86)\WaveMetrics\Igor Pro Folder (Windows 64-bit)
```

1. **Double-click the Igor Pro application file on your hard disk.**

   On Windows you can also start Igor using the Start menu.

   If Igor was already running, choose the File→New Experiment menu item.

2. **Use the Misc menu to turn preferences off.**

   Turning preferences off ensures that the tour works the same for everyone.

Load Data → Make a wave

Entering Data

1. **If a table window is showing, click in it to bring it to the front.**

   When Igor starts up, it creates a new blank table unless this feature is turned off in the Miscellaneous Settings dialog. If the table is not showing, perform the following two steps:

   1a. **Choose the Windows→New Table menu item.**

   The New Table dialog appears.

   1b. **Click the Do It button.**

   A new blank table is created.

2. **Type "0.1" and then press Return or Enter on your keyboard.**

   This creates a wave named "wave0" with 0.1 for the first point. Entering a value in the first row (point 0) of the first blank column automatically creates a new wave.

3. **Type the following numbers, pressing Return or Enter after each one:**

   1.2
   1.9
   2.6
   4.5
   5.1
   5.8
   7.8
   8.3
   9.7

Your table should look like this:

Table0:wave0

R10

Point	wave0
0	0.1
1	1.2
2	1.9
3	2.6
4	4.5
5	5.1
6	5.8
7	7.8
8	8.3
9	9.7
10	

4. **Click in the first cell of the first blank column.**
5. **Enter the following numbers in the same way:**

   -0.12
   -0.08
   1.3
   1
   0.54
   0.47
   0.44
   0.2
   0.24
   0.13

6. **Choose Data→Rename.**
7. **Click "wave0" in the list and then click the arrow icon.**
8. **Replace "wave0" with "time".**

   Notice that you can't use the name "time" because it is the name of a apologize for usurping such a common name.

9. **Change the name to "timeval".**
10. **Select "wave1" from the list, click the arrow icon, and type "yval".**
11. **Click Do It.**

Plot a Wave

Making a Graph

1. **Choose the Windows→New Graph menu item.**

   The New Graph dialog will appear. This dialog comes in a simple form that most people will use and a more complex form that you can use to create complex multiaxis graphs in one step.

2. **If you see a button labeled Fewer Choices, click it.**

   The button is initially labeled More Choices because the simpler form of the dialog is the default.

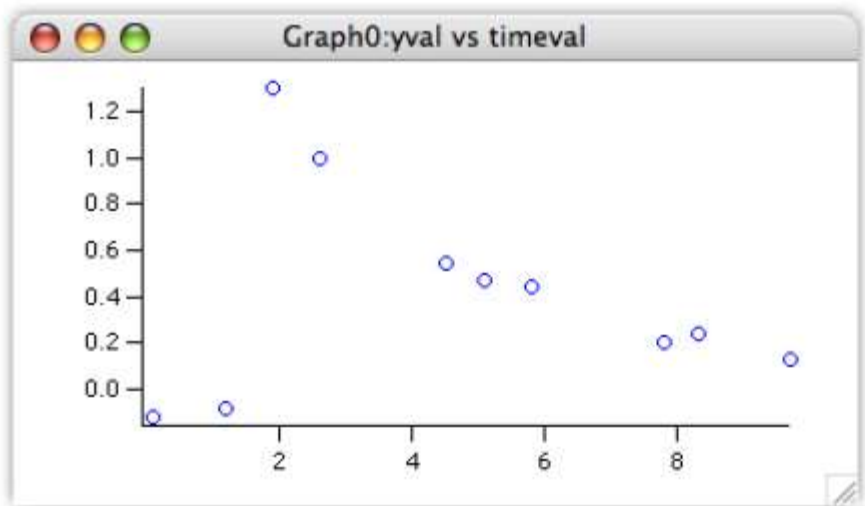
3. **In the Y Wave(s) list, select "yval".**

4. **In the X Wave list, select "timeval".**

5. **Click Do It.**

   A simple graph is created.

Your graph should now look like this:



6. **Position the cursor over the bottom axis line.**

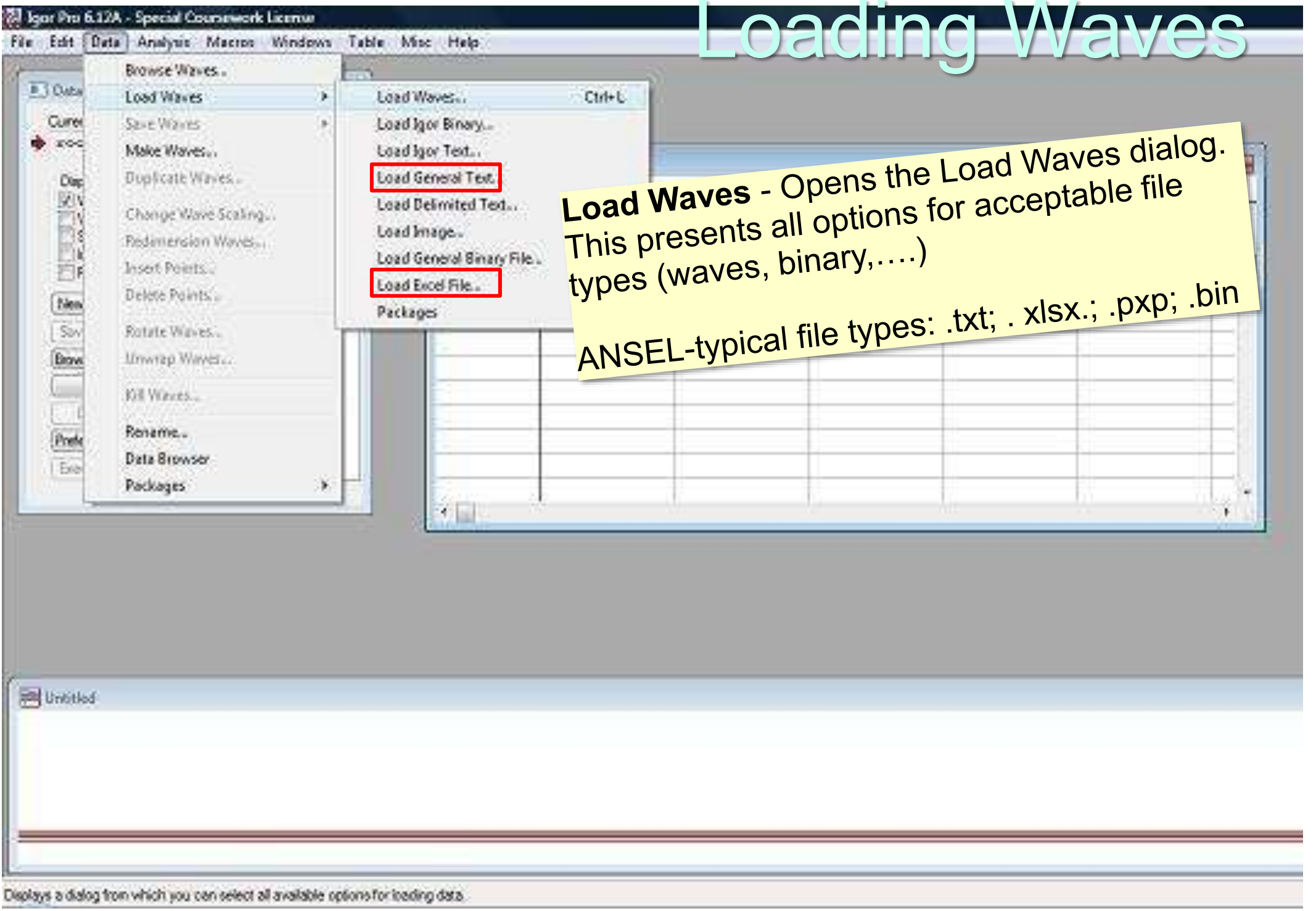
   The cursor changes to this shape: ↕. This indicates the cursor is over the axis and also that you can offset the axis (and the corresponding plot area edge) to a new position.

7. **Double-click directly on the axis.**

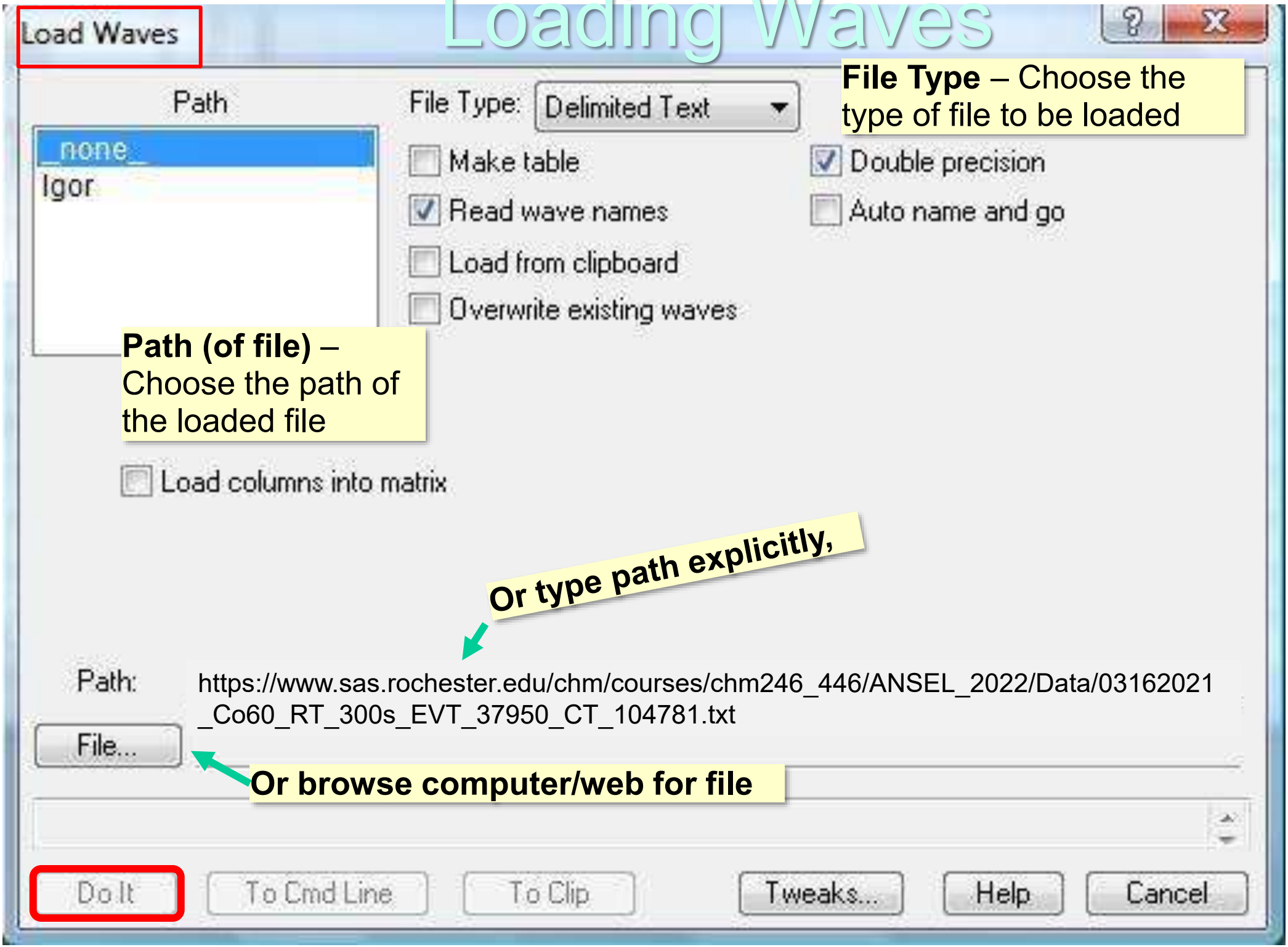
   The Modify Axis dialog appears. If another dialog appears, click cancel and try again, making sure the ↕ cursor is showing.

   Note the Live Update checkbox in the top/right corner of the Modify Axis dialog. When it is checked, changes that you make in the dialog are immediately reflected in the graph. When it is unchecked, the changes appear only when you click Do It. The Modify Axis dialog is the only one with a Live Update checkbox.

Loading Waves



Loading Waves



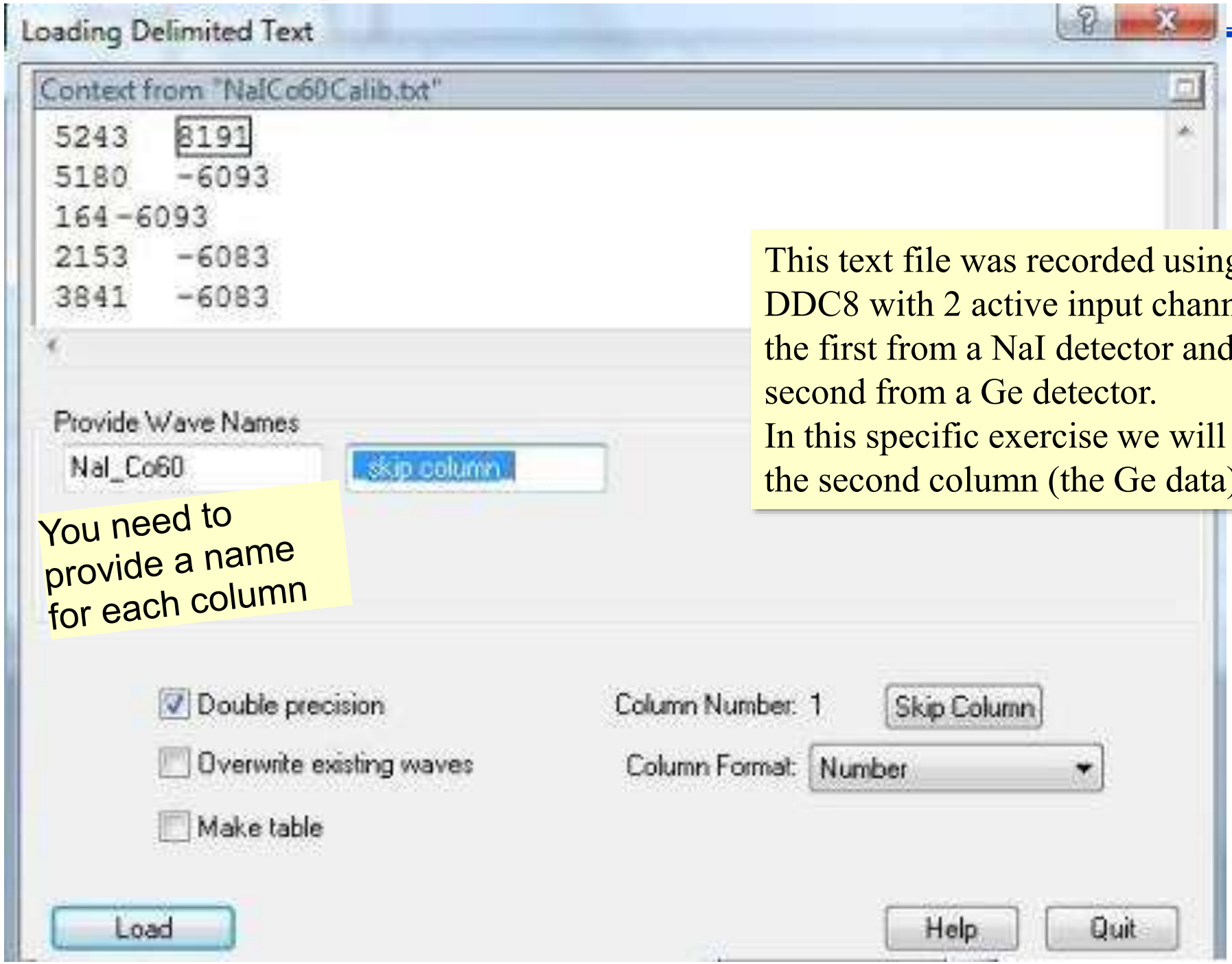
Loading Data

Summary: The basic sequence of actions to load 1D data, encoded as a delimited-text file, into IGOR

1. Bring up the ***Load Waves*** dialog.
2. Choose ***Load General Text*** *or* ***Load Delimited Text*** from the File Type pop-up menu.
3. Click the ***File*** button to select the file containing the data.
4. Click ***Do It***.

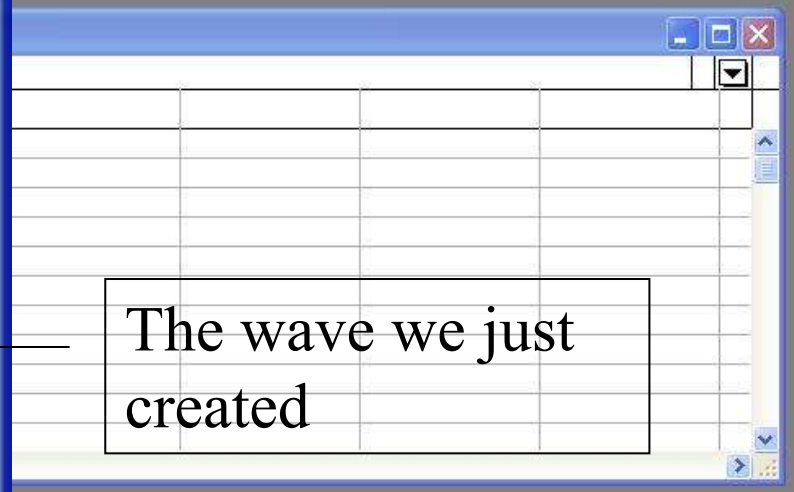
Only after you click *Do It*, does the *Load Wave* operation run and execute the *Load General/Delimited Text* routines.

Loading Delimited-Text Files



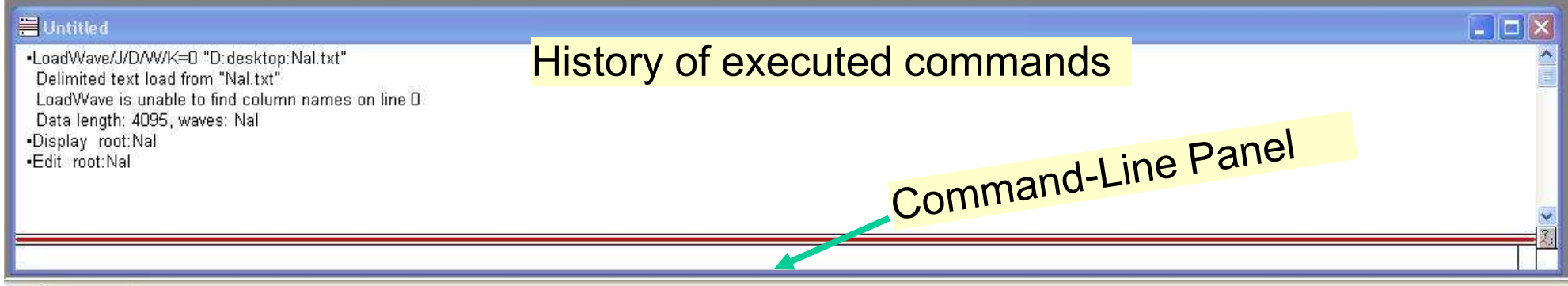
This text file was recorded using the DDC8 with 2 active input channels: the first from a NaI detector and a second from a Ge detector.
In this specific exercise we will skip the second column (the Ge data).

Browse for Loaded Wave



The wave we just created

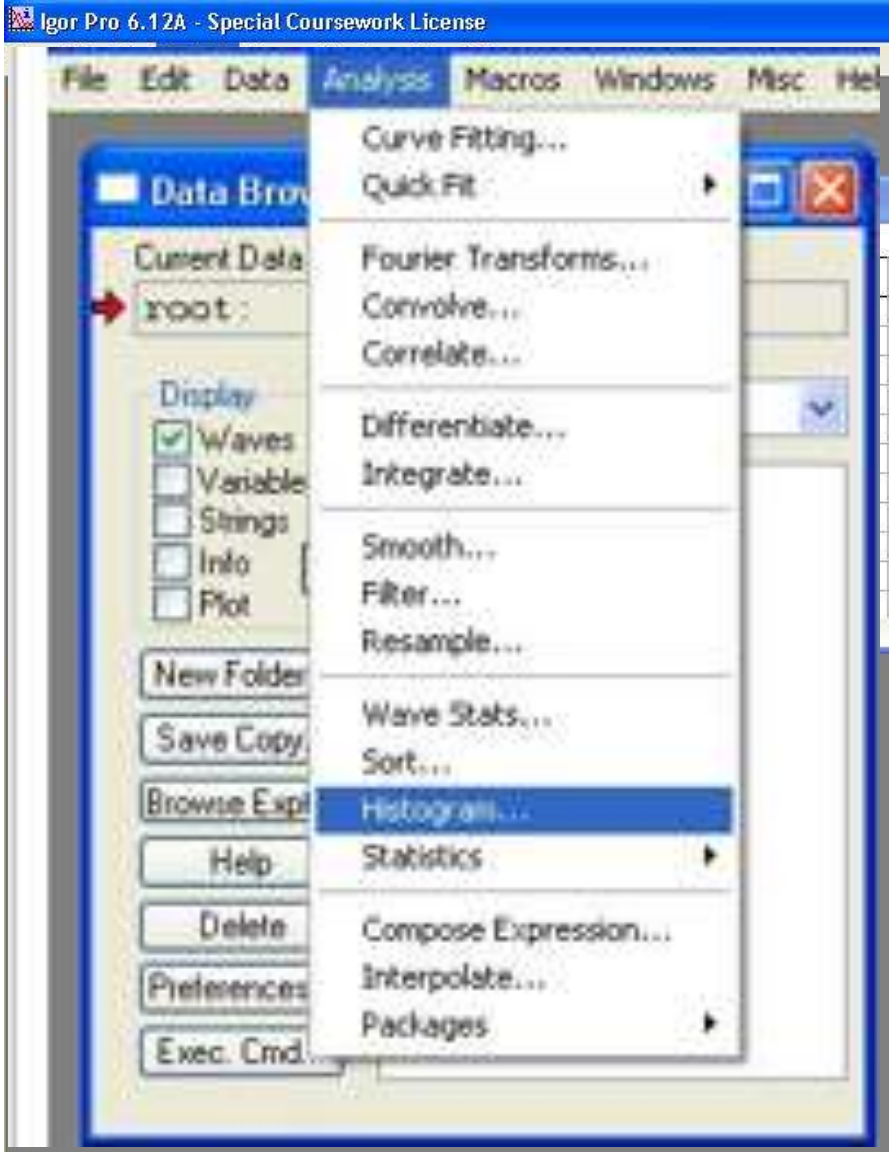
The uploaded wave will appear in the Data Browser under the name you gave the column.

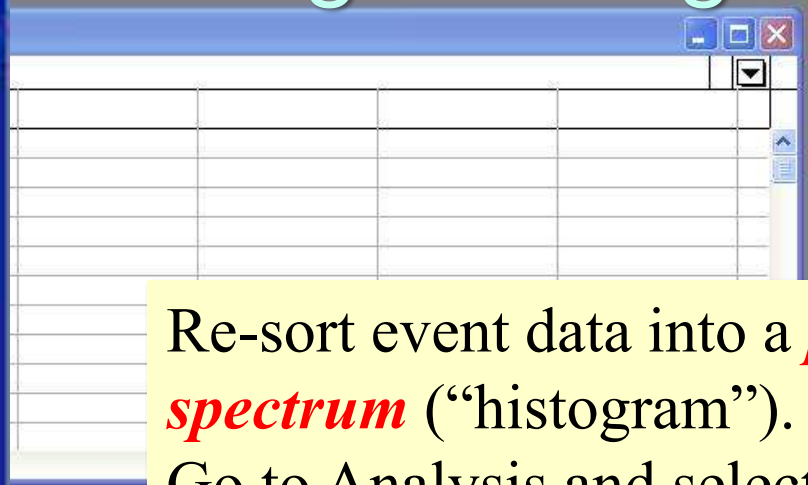


History of executed commands

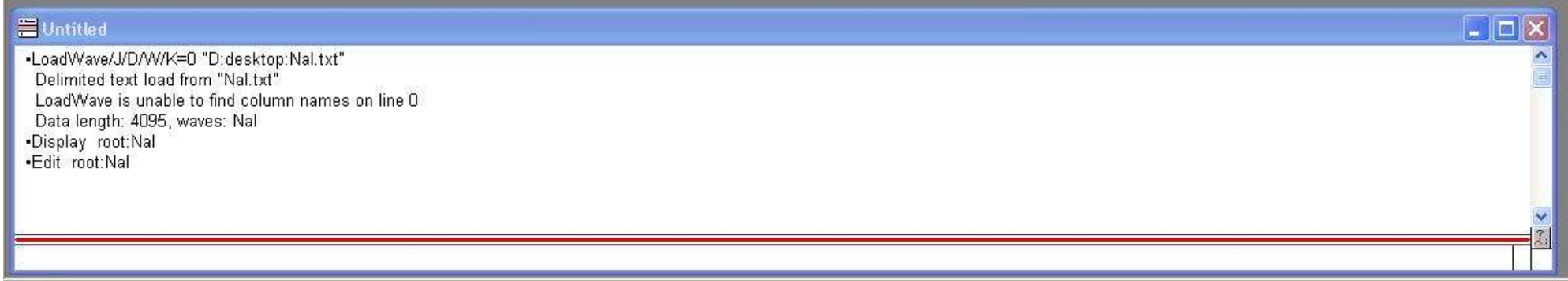
Command-Line Panel

Histogramming/Sorting

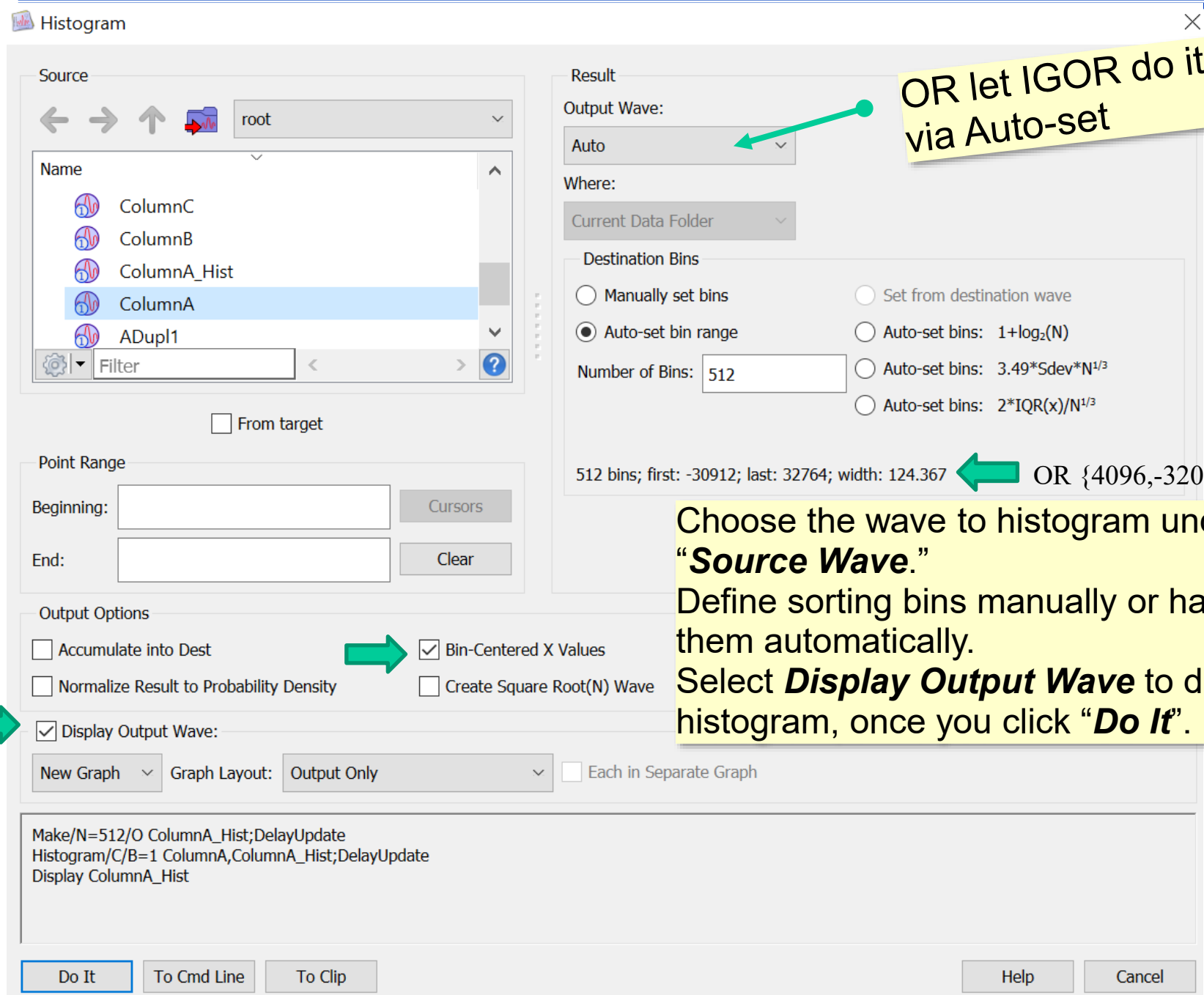




Re-sort event data into a ***probability spectrum*** ("histogram").
Go to Analysis and select Histogram



Histogramming Event Data



Choose the wave to histogram under "***Source Wave***."
Define sorting bins manually or have Igor set them automatically.
Select ***Display Output Wave*** to display the histogram, once you click "***Do It***".

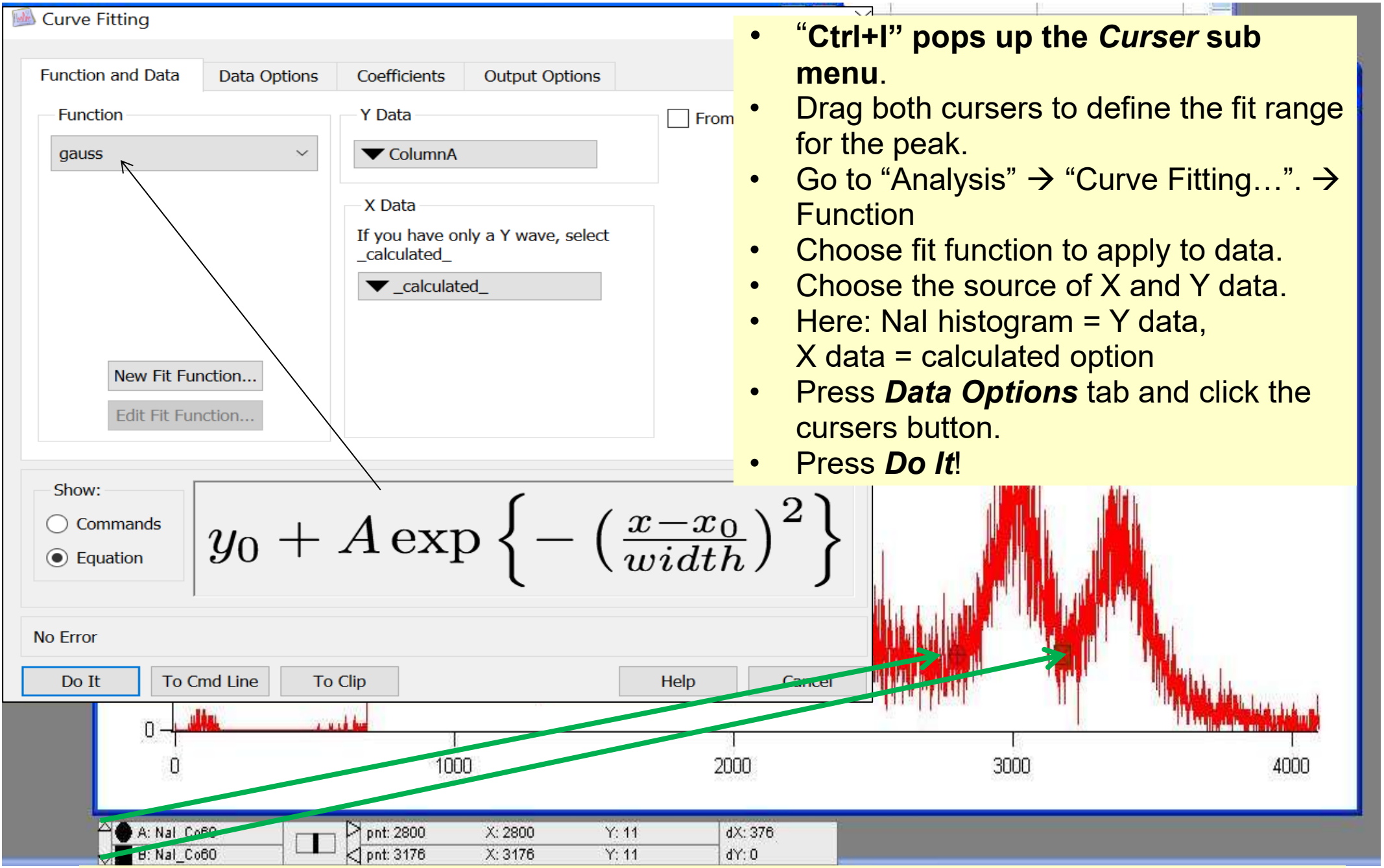
Displaying and Editing Histograms

- If you chose not to display the histogram from the histogram menu, you can right-click the histogram name and click display.

- Once a histogram is displayed you can edit the figure.
- Double clicking on one of the axes pops up the “Modify Axis” menu.
- You can set the range of each axis, label them and do other fancy things.

- Right-clicking the middle of the figure pops a menu that lets you append traces to the graph, add an annotation and edit other features of the figure.

Analysis – Fitting a Curve to Data

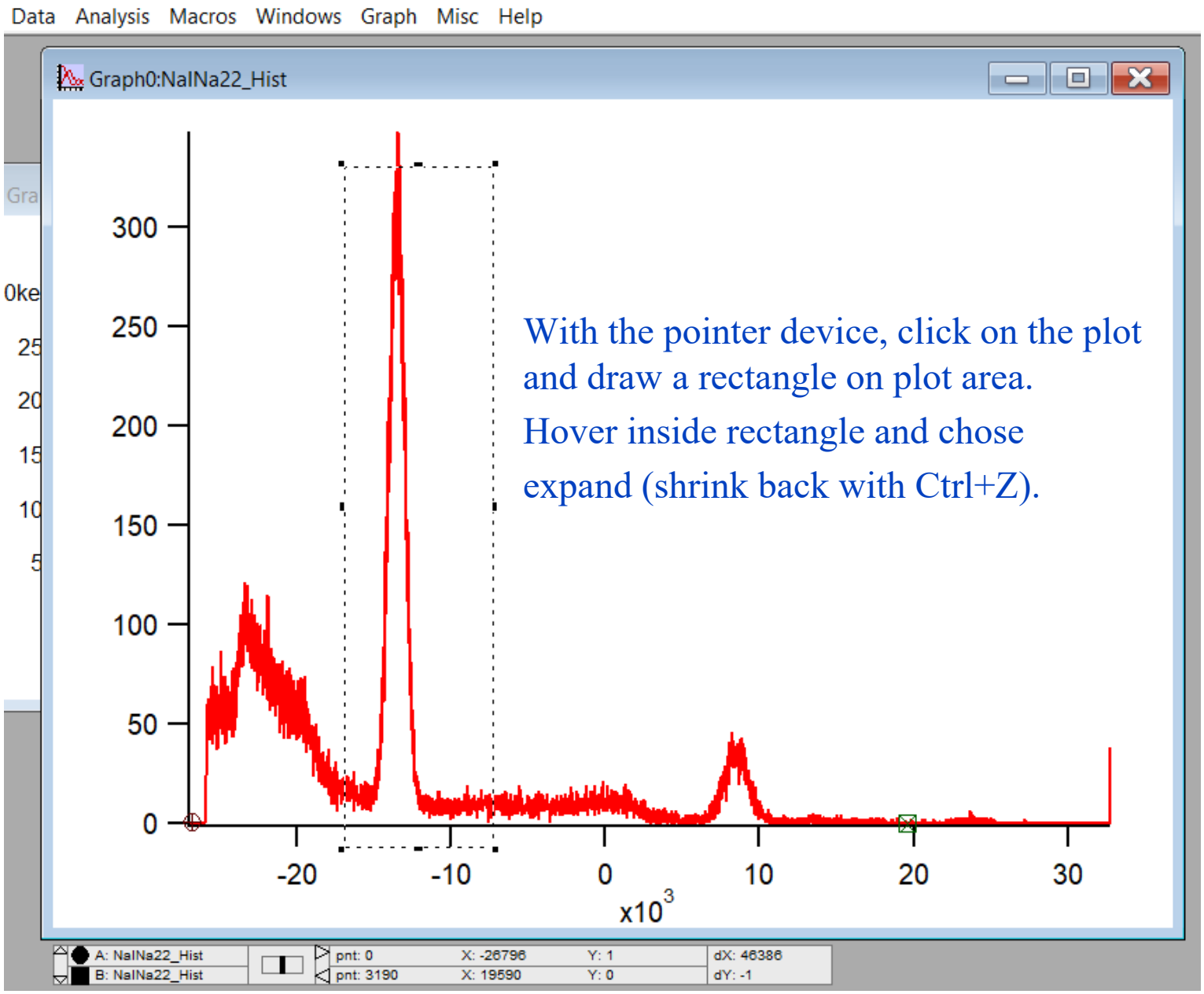


- **"Ctrl+I" pops up the *Curser* sub menu**.
- Drag both cursers to define the fit range for the peak.
- Go to "Analysis" → "Curve Fitting…". → Function
- Choose fit function to apply to data.
- Choose the source of X and Y data.
- Here: NaI histogram = Y data, X data = calculated option
- Press ***Data Options*** tab and click the cursers button.
- Press ***Do It***!

A fit trace should appear in the range between the cursers. The command box below shows values of each parameter and its error.

Expand Plot Regions Na-22 Data

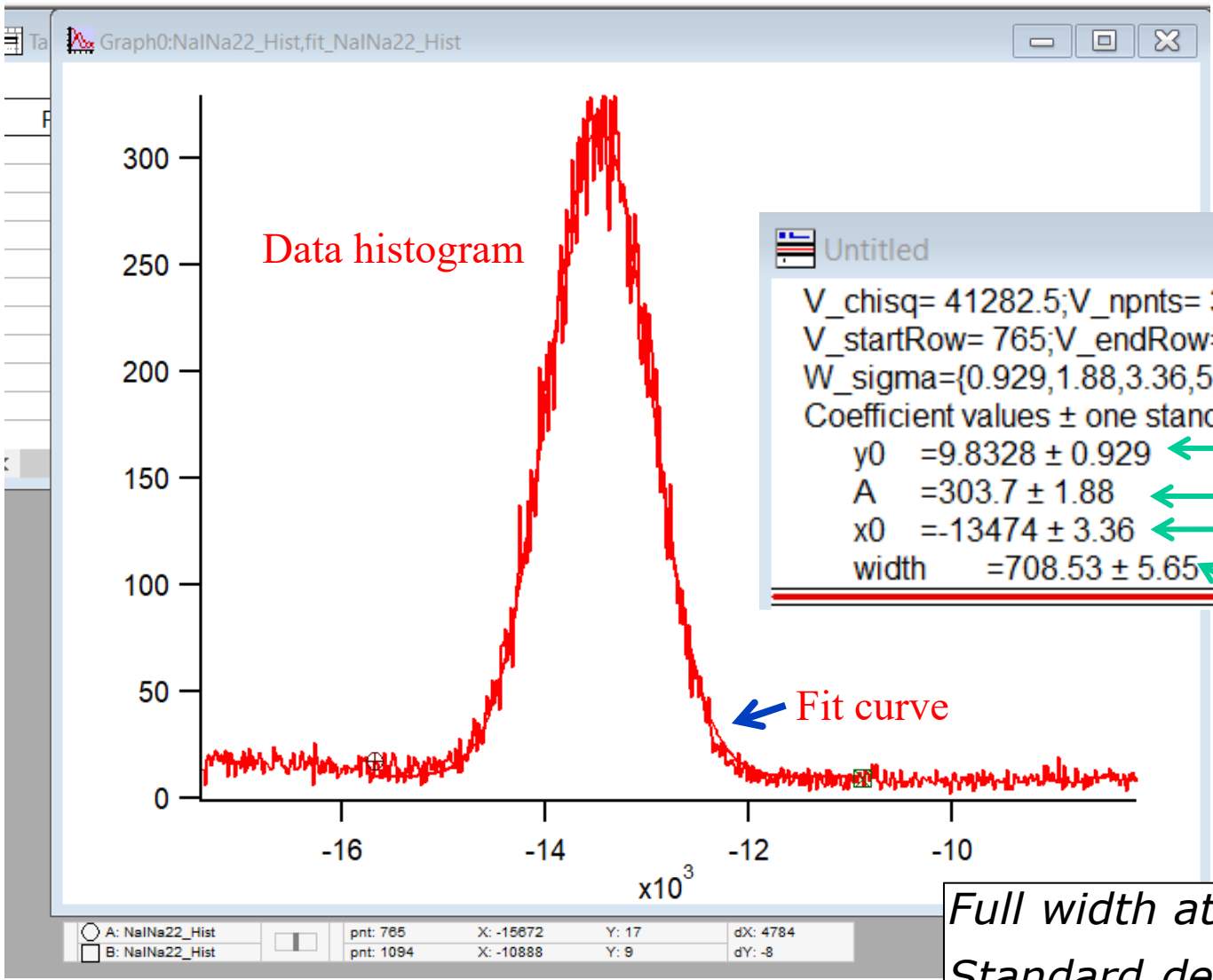
Na-22 spectrum has well separated lines:, 511-keV γ line



Analysis-Gauss Fit to Data

Na-22 spectrum, 511-keV γ line

Draw rectangle on plot area, expand (back with Ctrl=Z)



Untitled

V_chisq= 41282.5;V_npnts= 330;V_numNaNs= 0;V_numINFs= 0;
V_startRow= 765;V_endRow= 1094;
W_sigma={0.929,1.88,3.36,5.65}
Coefficient values ± one standa

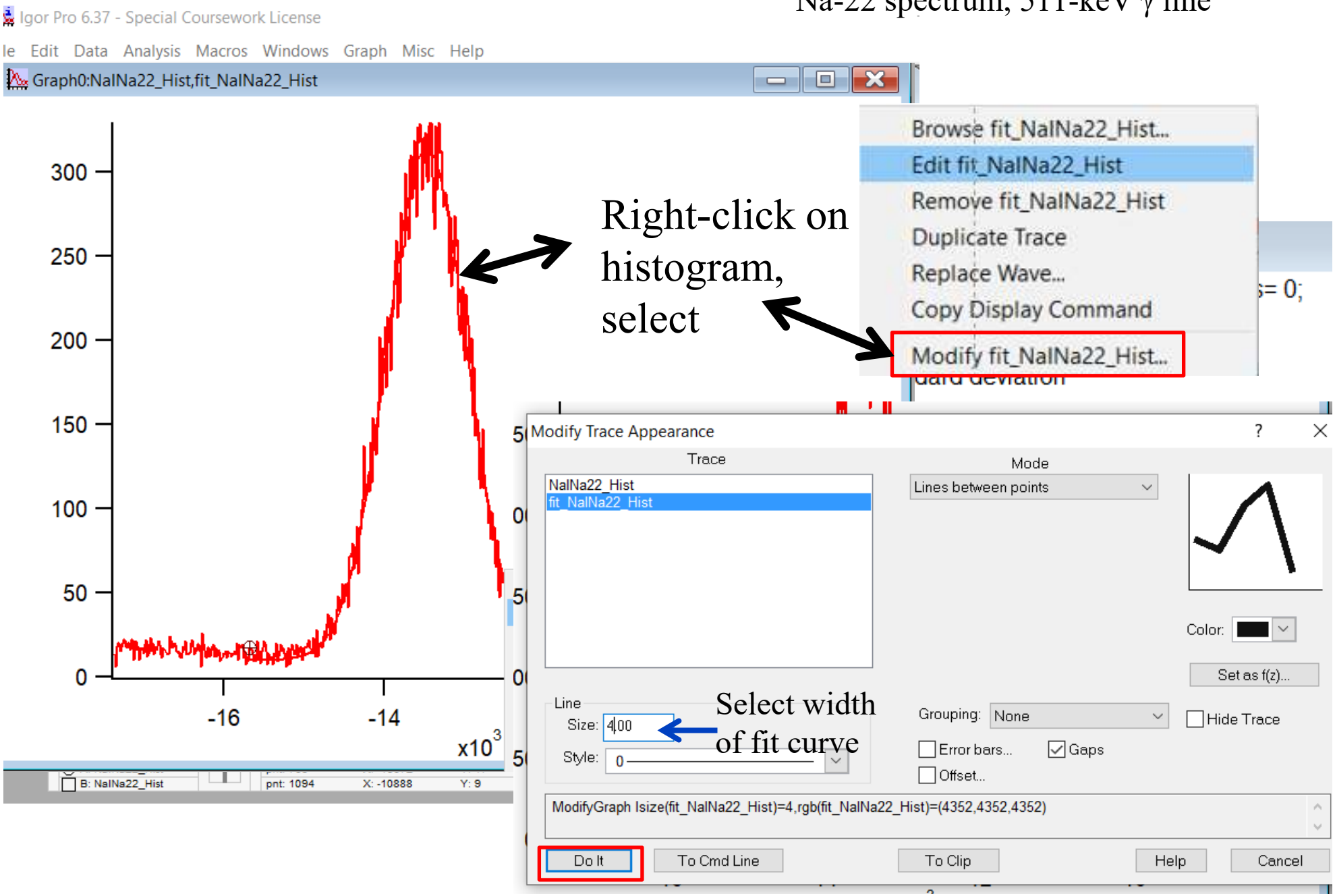
y0	=9.8328 ± 0.929	← Background intensity
A	=303.7 ± 1.88	← Line area
x0	=-13474 ± 3.36	← Center of line
width	=708.53 ± 5.65	← Width of line (see definition)

Full width at half maximum : Γ_{FWHM}

Standard deviation : $\sigma = \Gamma_{FWHM}/3.54$

Analysis: Gaussian Fit to Data

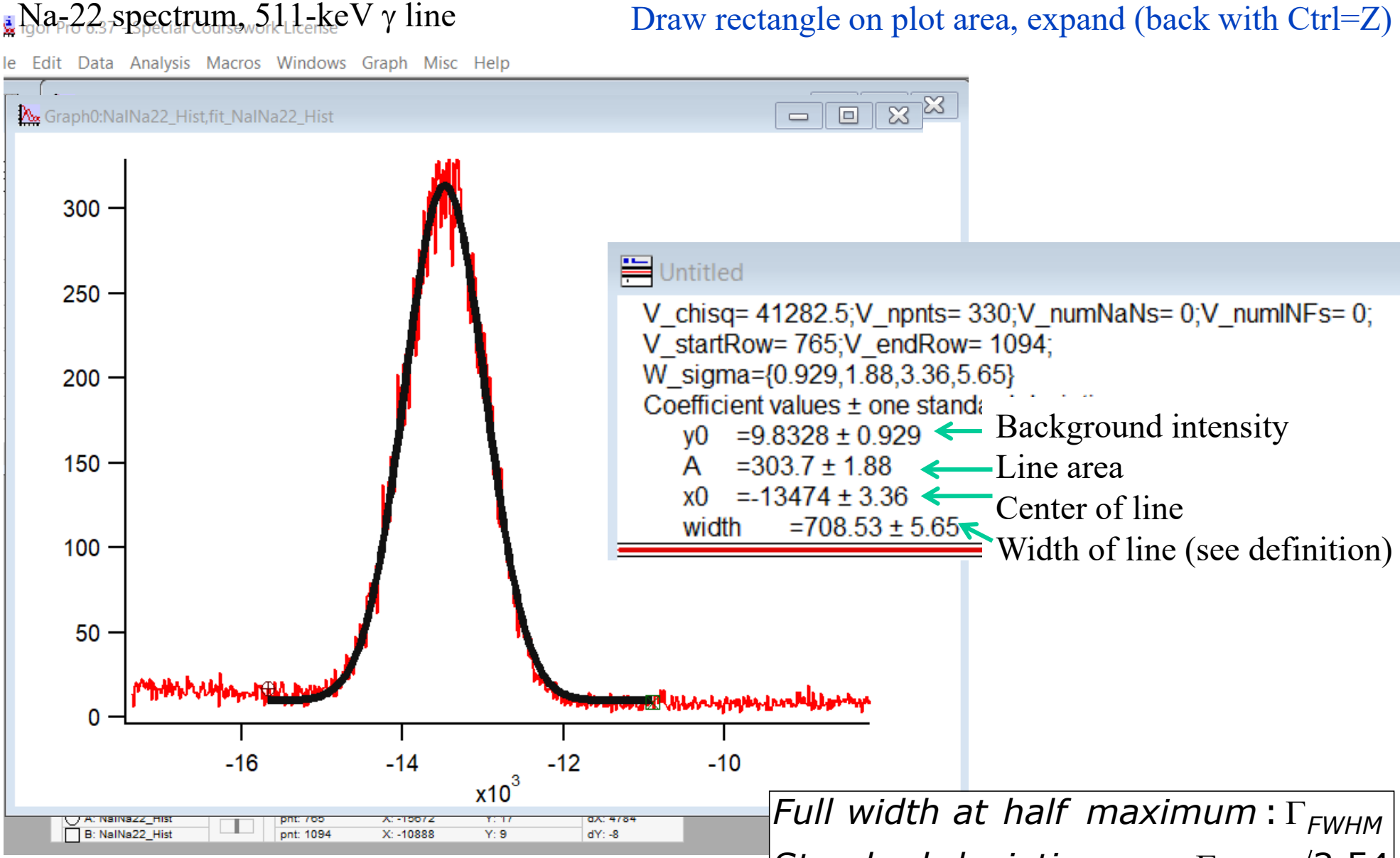
Na-22 spectrum, 511-keV γ line



Analysis-Gaussian Fit to Data

Na-22 spectrum, 511-keV γ line

Draw rectangle on plot area, expand (back with Ctrl=Z)



Full width at half maximum : Γ_{FWHM}

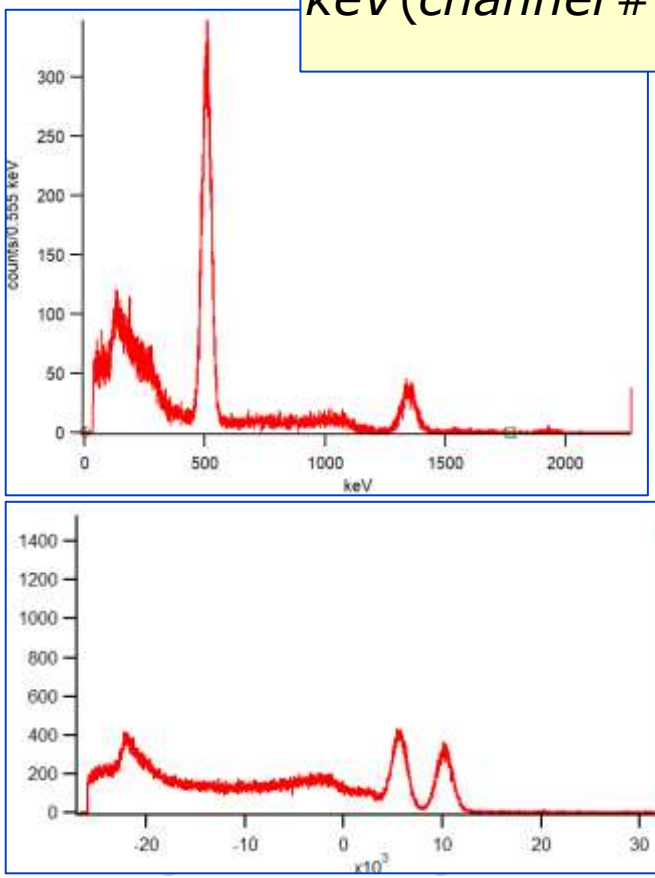
Standard deviation : $\sigma = \Gamma_{FWHM}/3.54$

Perform an Energy Calibration of Detector

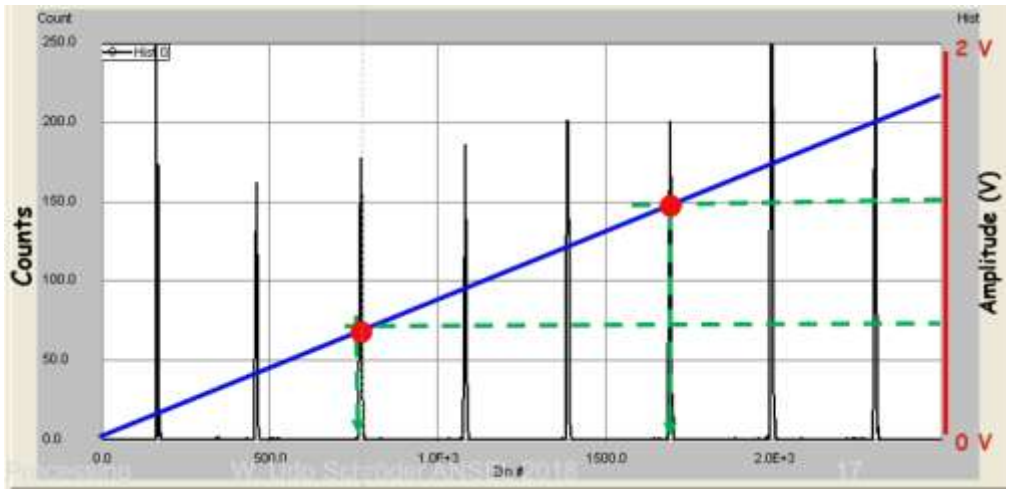
Measure spectrafor various calibration sources with the same detector. If necessary, extend or interpolate gaps with normalized pulse generator (Pulser). Fit relation between line channel #s and known line energies → obtain linear calibration curve. Keep track of errors in calibration constants.

$$Energy = Energy(channel\,\#)$$

$$keV(channel\,\#) = offset(keV) + \left(\frac{keV}{channel\,\#}\right) \cdot channel\,\#$$

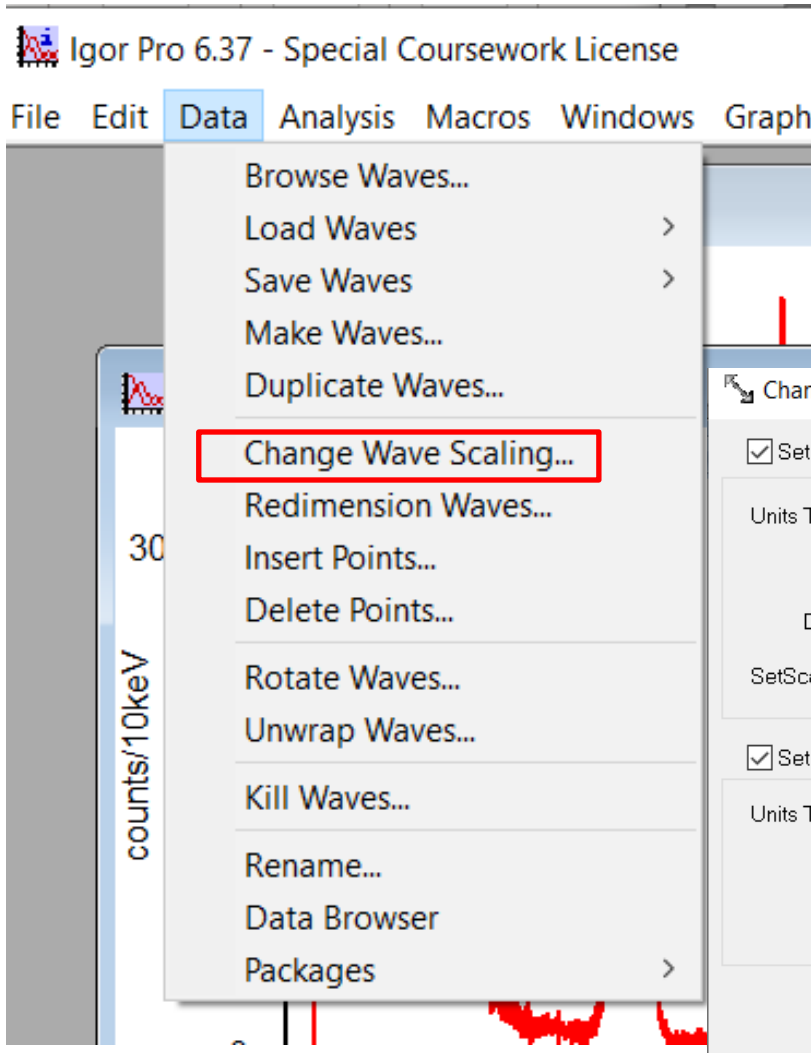


"Pulser Fence" calibrates abscissa of DAQ (Channel, Bin#) in Volts, MeV, etc.



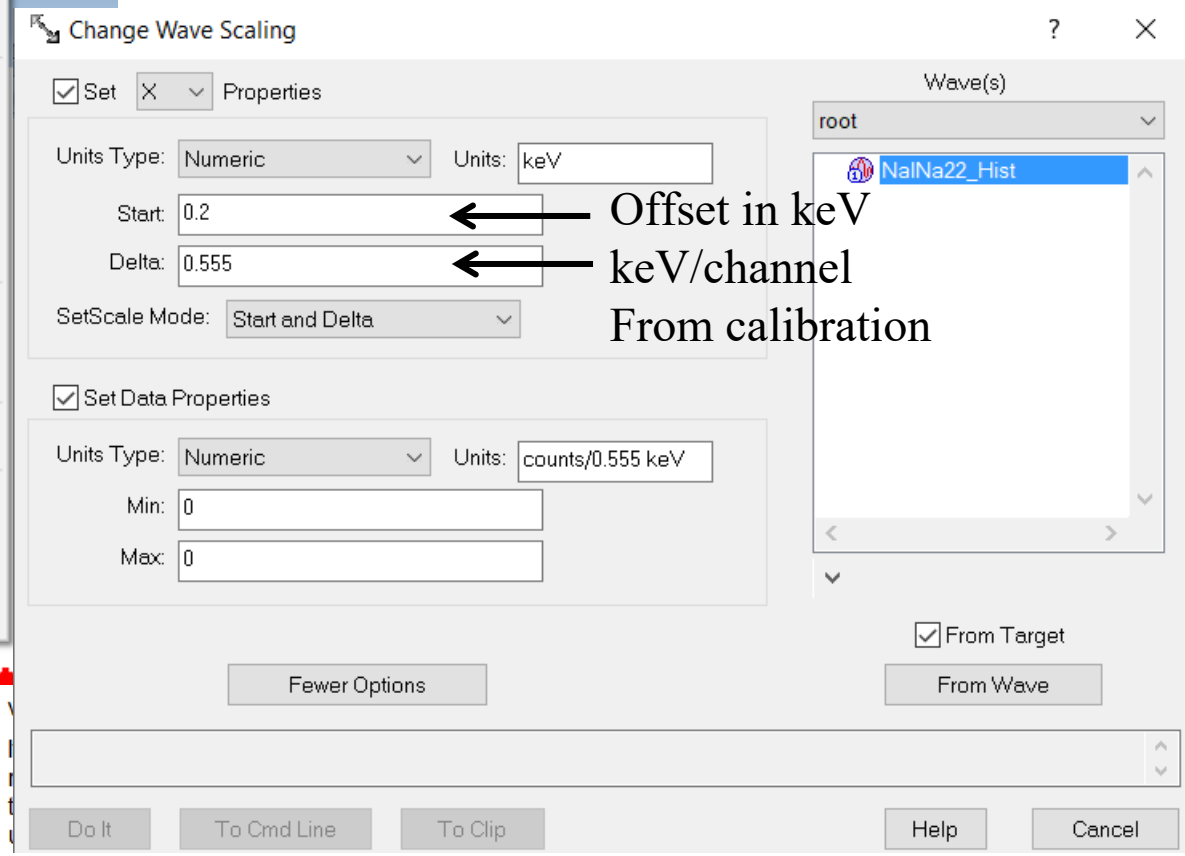
Analysis: Energy Calibrate Data

Na-22 spectrum, 511-keV γ line, following is a rough guess of energy scale of channel#s



Procedure window shows:

.SetScale/P*0.2,9,.555,"keV",NaINa22-dupl



Offset in keV

keV/channel

From calibration

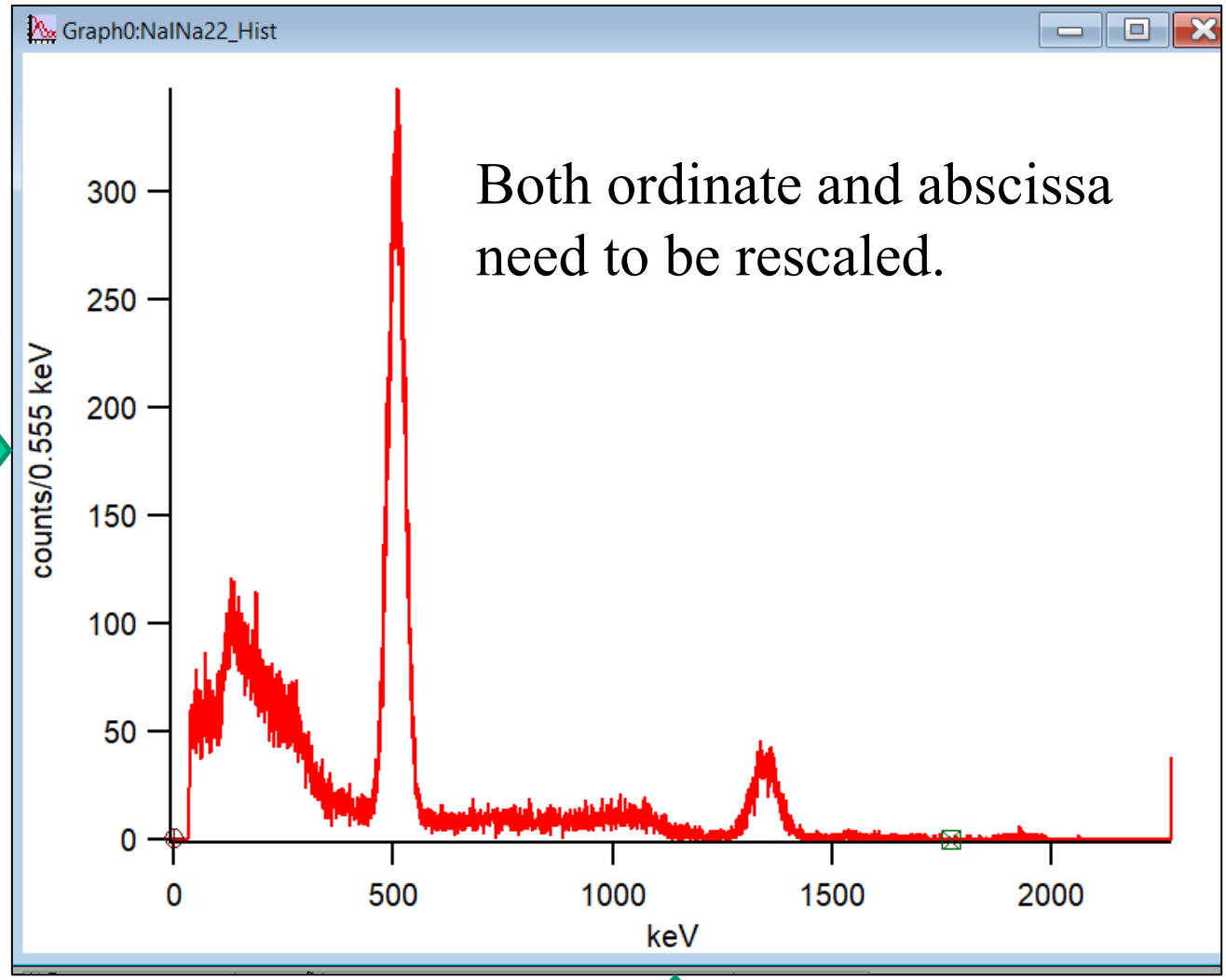
Analysis-Energy Calibrate Data

Na-22 spectrum, 511-keV γ line, following is a rough guess of energy scale of channel#s

1 channel=0.555keV

Number of counts/channel is now counts/(channel width in units of keV).

Rescale to counts/keV !



Changed scale to keV

(γ line is roughly at the correct energy, but not exactly)

User Defined Functions

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 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

User Defined Functions - Example

Type in the procedure window the following:

```
FUNCTION FitGauss (wav, from,to)
VARIABLE from,to
WAVE wav
    CurveFit/Q gauss wav(from,to) /D
    VARIABLE PeakPos, sig, FWHM, PeakArea
    WAVE W_Coef = W_Coef
    PeakPos = W_Coef[2]
    FWHM = W_Coef[3]*2*sqrt(ln(2))
    sig  = W_Coef[3] / sqrt(2)
    PRINT "  pos, sigma, fwhm =", PeakPos, sig, FWHM, "
    DC pedestal=", W_Woef[0]
END
```

User Defined Functions – Better Example

Code by Prof. Frank Wolfs

```
Function SingleGauss (w, x) : FitFunc

    // Define fit function - single Gaussian
    // We assume there are 5 parameters, which are stored in the array w:
    //
    // Background paramater:          background = w[0] * x + w[1]
    //                                w[0] = slope
    //                                w[1] = offset
    //
    // Gaussian paramters:            double Gaussian = w[2]*exp(-(x-w[3])^2/(2*w[4]^2))
    //                                w[2] = amplitudes peak 1
    //                                w[3] = centroids peak 1
    //                                w[4] = sigma peak 1

    // Parameter declaration
    VARIABLE x                              // X coordinate
    WAVE w                                  // Wave with the coefficient of the fit

    // Local variable declaration
    VARIABLE bg                 // Background
    VARIABLE peak1              // Gaussian # 1

    // Calculate the background
    bg = w[0]*x + w[1]

    // Calculate the peaks
    peak1 = w[2]*exp(-(x-w[3])^2/(2*w[4]^2))

    // Return the function value
    return bg + peak1

End.
```

User Defined Gaussian Fit

- After compiling the user defined function, you can again place the cursers in the peak range you want to fit.
- Type "FitGauss (name of the fitted wave, xcsr(A), xcsr(B))" in the command bar in the command window.
- After you press enter the command window will process the fit and give you the position of the peak (pos), the width of the peak (sigma) and the full width at half maximum (FWHM).
- The user can create all sorts of functions to assist his/her analysis, such as performing multiple gaussian fits, linear fits, more background functions, etc..