Tutorials for Origin 8.5 SR0

OriginLab Corporation

One Roundhouse Plaza

Northampton, MA 01060

USA

(413) 586-2013

(800) 969-7720

Fax (413) 585-0126

www.OriginLab.com

Table of Contents

1 Introduction Tutorial

Welcome to the Origin Version 8.5 Tutorial Guide

The material in this guide is designed to provide both new and advanced users with specific instructions on how to perform the most commonly used and powerful features in Origin. If you are a new user, or would simply like to get acquainted with the Origin8 user interface, the first lesson in this manual covers the **Origin GUI** (graphical user interface). It introduces the basic concepts involved in manipulating workbooks, creating graph windows and managing workspace with Project Explorer. The other tutorials in this guide handle much more specific tasks, so we recommend you look through them at your leisure as you find you need some pointers on specific operations.

A general note before proceeding:

> You will find references to buttons found on various toolbars in many of the tutorials in this guide. These buttons are shortcuts to menu commands. If you don't see the button referenced in a tutorial, it may simply not be shown in your workspace. To open a toolbar, select **View: Toolbars**, click on the checkbox next to the desired toolbar, and then click **Close**.

2 User Interface

This chapter contains tutorials that pertain to the Origin Graphical User Interface.

- Origin GUI

2.1 Origin GUI

2.1.1 Summary

This tutorial will introduce you to the Origin workspace. You will learn about the different kinds of Origin Windows that make up a Project, and how to manipulate these windows with Project Explorer.

Minimum Origin Version Required: Origin 8.0 SR6

2.1.2 What you will learn

This tutorial will show you how to:

- Manipulate Workbooks
- Create Graph Windows
- Manage workspace with Project Explorer

2.1.3 Projects

The Origin Project File is a convenient container for all of your data whether it be loose or associated with child windows - workbooks, graphs, and matrices. It also contains results saved in the Results Log.

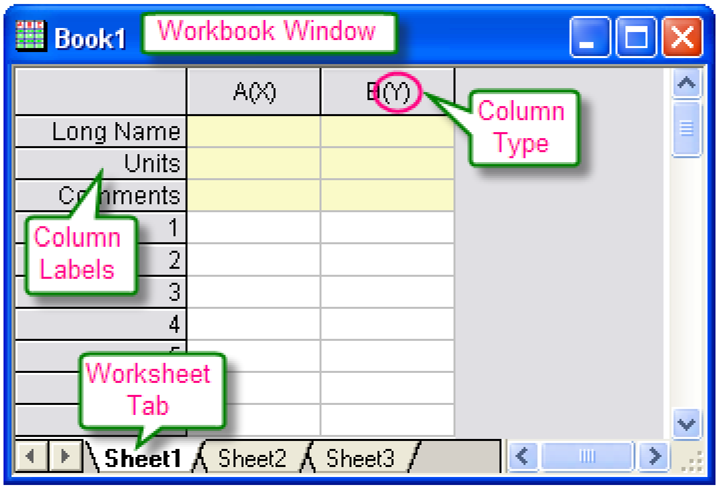
Only one project file can be open but you can append the contents of one project onto another.

2.1.4 Windows

Origin has numerous windows and workspaces available for completing various tasks. You can see all types of windows from the **New** dialog (**File: New**). The most frequently used windows are **Workbook**, **Graph**, and **Matrix**.

Workbook

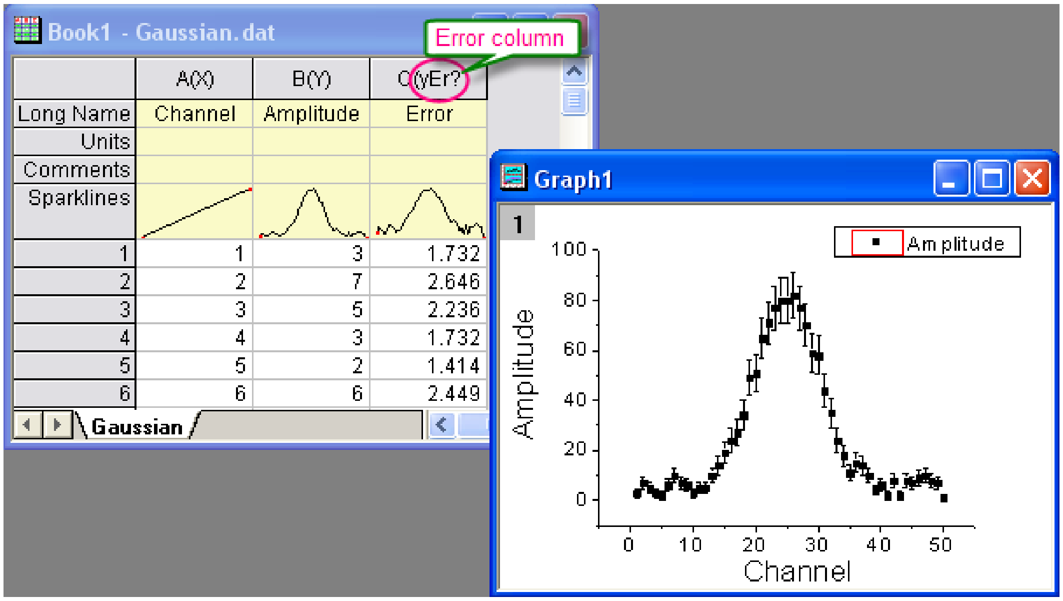
The Origin workbook is the primary structure for organizing your data. Each workbook is composed of one or more Origin worksheets. And each worksheet, in turn, is composed (usually) of one or more worksheet columns or datasets. Columns in Origin have different types, such as X, Y, Z, yError, etc, which represent the plot designation for graphing.



To learn how to manipulate Origin workbooks, try the following:

1. Select **File: New** from the menu and choose *Workbook* to create a new workbook.
2. Select **File: Import: Single ASCII** to bring up the **Open** dialog. Browse to the *Samples**Curve Fitting* subfolder of the Origin program folder. Highlight the file *Gaussian.dat* and click the **Open** button to import the data into the Origin worksheet.
3. On import, sparklines were automatically turned on, allowing you to quickly view the shape of the data; the sheet name became the name of the file; and as needed an additional column was added to the worksheet. You can see from the **Long Name** that the 3rd column represents data error. To set this column as an Error Column, click the column title to highlight it, right-click to bring up a fly-out menu, and then select **Set As: Y Error**.

4. Plotting data in Origin is now easy; highlight all three columns and select **Plot: Symbol: Scatter** from the menu to create a scatter plot.



	A(X)	B(Y)	C(yEr?)
Long Name	Channel	Amplitude	Error
Units			
Comments			
Sparklines			
1	1	3	1.732
2	2	7	2.646
3	3	5	2.236
4	4	3	1.732
5	5	2	1.414
6	6	6	2.449

Graph

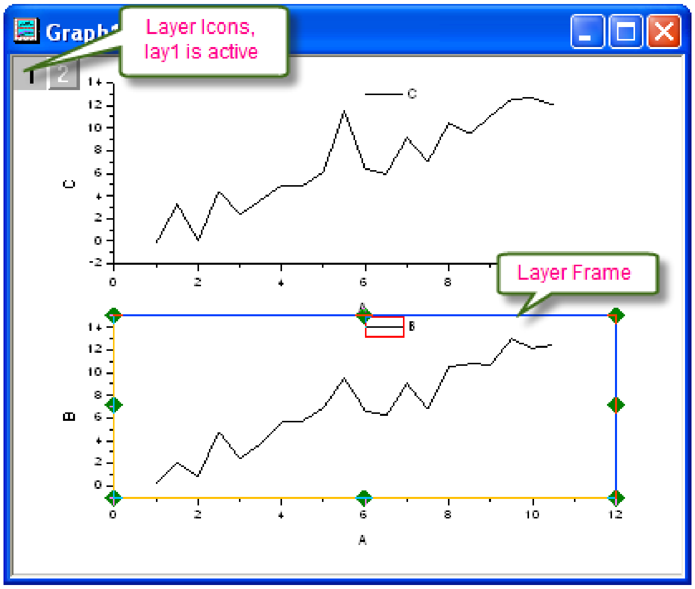
The Graph window is a container for graphical depictions of your experimental data and your analysis results. Graph windows may contain a single plot in a single graph layer or they may contain multiple plots in multiple graph layers.

The graph layer is the fundamental unit of the Origin graph. The layer is comprised of a set of axis scale values, one or more data plots, and any included text labels, drawing objects, graph legends/color scales, button objects, etc. Graph layers can be created, sized, and moved independently of one another, allowing you a great deal of latitude in charting your data.

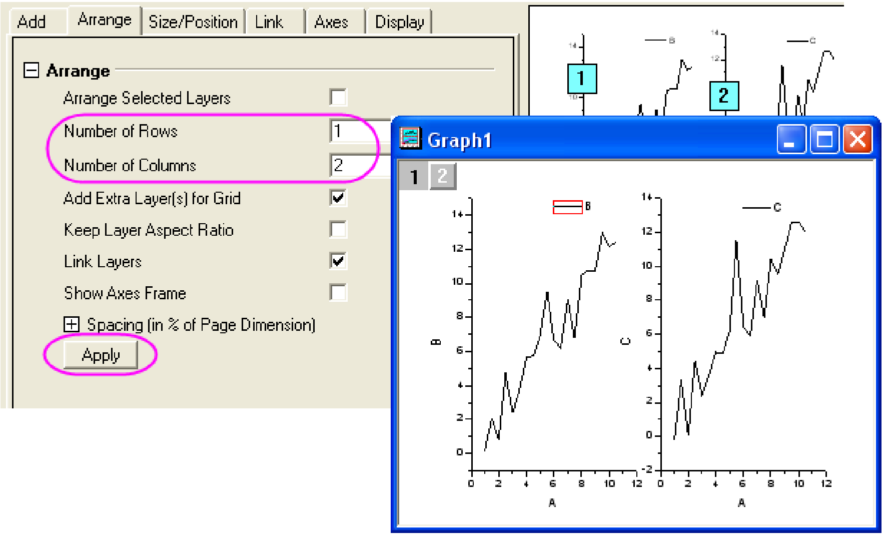
To learn how to organize layers in a graph window, try the following:

1. Create a new workbook, and import the file *Linear Fit.dat* from the *Sample**Curve Fitting* folder. You can see that there are three Y columns and one X column after import; each Y column will use the left-most X column as its X coordinates.

2. Highlight columns B and C, and select **Plot: Multi-Curve: Vertical 2 Panel** to plot the curves. This is a two layer graph. While a graph can have multiple layers, only one layer is active at any given time. You can perform operations on the active layer, such as resizing, changing the plot color, etc. When working on a layer, the active layer is denoted by a depressed layer *n* icon in upper left corner of the graph window.



3. To rearrange these two layers, with the graph window active, select the menu item **Graph: Layer Management** to bring up the dialog. Activate the **Arrange** tab, on the middle panel, enter 2 in the *Column* edit box; enter 1 in the *Row* edit box and click the **Apply** button. After you click the **OK** button, the graph layers are arranged horizontally.



⊟ Arrange

Arrange Selected Layers ☐

Number of Rows 1

Number of Columns 2

Add Extra Layer(s) for Grid ☑

Keep Layer Aspect Ratio ☐

Link Layers ☑

Show Axes Frame ☐

⊞ Spacing (in % of Page Dimension)

Apply

Most often you will use the worksheet for tabulating and manipulating your data, while you will use the graph window for plotting your data. However, if you are making 3D surface or contour plots of XYZ data, you will need to become familiar with another window type, the Origin Matrix.

Matrix

The Origin matrix window is a container for one or more Origin matrices. Each matrix window contains one or more matrix sheets, and each matrix sheet can contain one or more matrix objects. The matrix object itself, is a vector of Z values. These Z values are related to one another in the X and Y dimensions by their relative row and column positions in the matrix. Matrices are a precursor to constructing Origin's 3D graph types such as contour graphs and color-mapped surfaces, and since they are used in depiction and manipulation of 3D data, they are used by Origin in image processing and analysis. We will show you how to use the Origin Matrix to create a 3D plot in a later tutorial.

2.1.5 Project Explorer

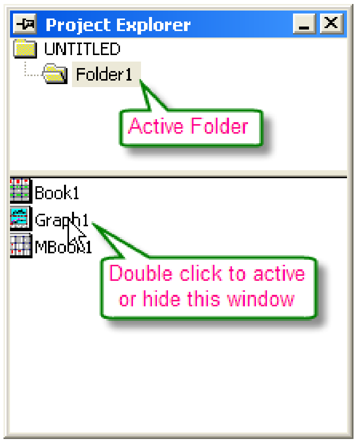
Typically, users amass quite a lot of data in an Origin project file. If you anticipate building a project file that contains dozens of worksheets, graphs, notes windows, etc., you will probably want to use Project Explorer (PE) to help you manage your Origin workspace. Project Explorer helps to organize your workspace so that you see only data that is relevant to the task at hand. In addition, you can use Project Explorer to create new project files from a portion of an existing project file or to append the contents of another project file to your current file. The Project Explorer workspace can be hidden or restored as needed.

Open/Close Project Explorer

When you first start Origin, Project Explorer displays docked to the edge of the workspace. You can dock it to any other edge or float it in the workspace. Because Project Explorer uses some of your workspace, you may want to close Project Explorer, even if you have already created a folder structure. To close/open Project Explorer, press **Alt + 1** or select menu item **View: Project Explorer**.

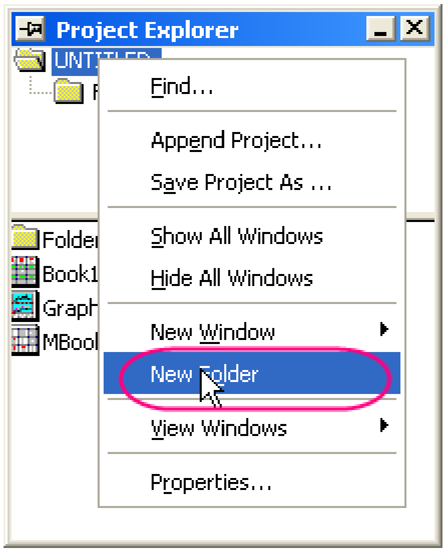
Browse Origin windows in Project Explorer

There are two panels in Project Explorer: the folders panel and the contents panel, which displays all objects in the active folder. When you start a new Origin session, you can click the new workbook , new graph or new matrix button to create some blank windows, and then you can see these window's icons in the contents panel. Double-click the icon to hide/view the window.



Add a subfolder in Project Explorer

To create a new folder, right-click on the project folder (or a subfolder) in the folder panel, and select **New Folder** from the shortcut menu.



Once you have created one or more subfolders, you can move child windows between folders by dragging & dropping them within the Project Explorer workspace.

3 Importing

- Simple ASCII
- Import Wizard
- Import Time Data
- Post Processing with Import Filter

3.1 Single ASCII

3.1.1 Summary

The **File: Import: Single ASCII** menu allows you to automatically import a single ASCII file where the data columns are delimited orderly and it consists of few header lines (maybe just a short description for the file and then names and units for the columns).

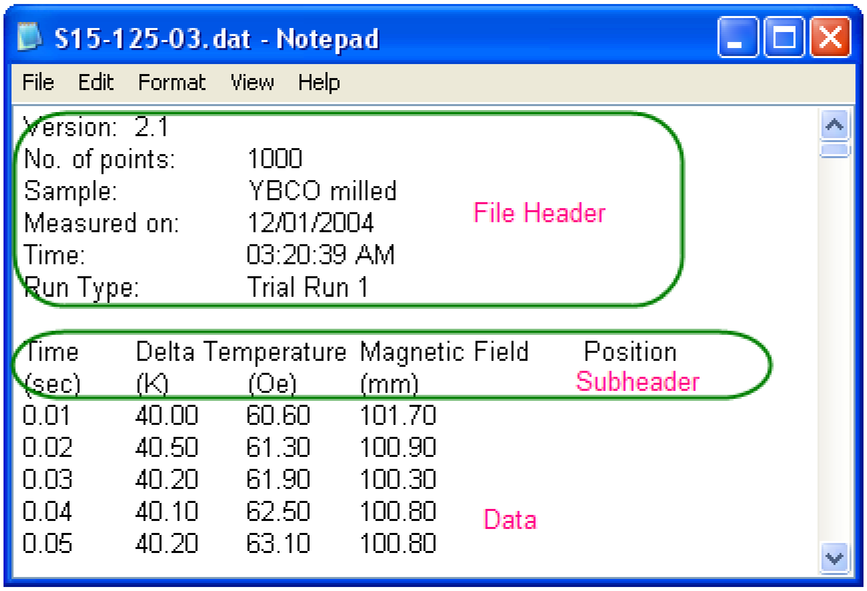
Minimum Origin Version Required: Origin 8.0 SR6

3.1.2 What you will learn

This tutorial will show you how to Import ASCII files.

3.1.3 Steps

Using Windows Explorer, browse to the \Samples\Import and Export subfolder of the Origin program folder (by default installed in the Program Files folder). Open the file *S15-125-03.dat* in Windows Notepad. You can see that this file includes header lines and data lines. For Single ASCII files, Origin can auto detect file header/subheader and extract this information to the worksheet headers, such as *Long Name*, *Units*, etc.



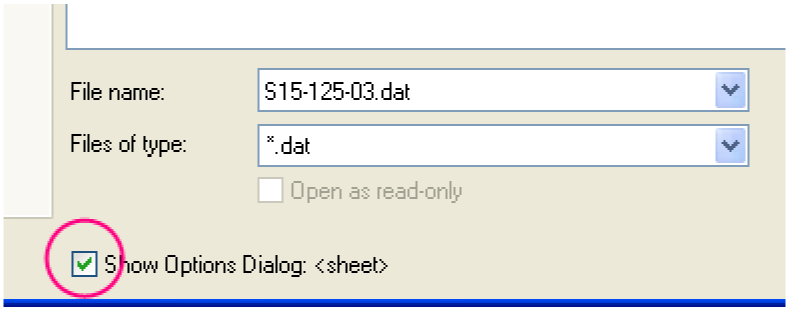
```
Version:  2.1
No. of points:      1000
Sample:             YBCO milled
Measured on:        12/01/2004
Time:               03:20:39 AM
Run Type:           Trial Run 1

Time      Delta Temperature  Magnetic Field     Position
(sec)     (K)           (Oe)          (mm)
0.01      40.00         60.60         101.70
0.02      40.50         61.30         100.90
0.03      40.20         61.90         100.30
0.04      40.10         62.50         100.80
0.05      40.20         63.10         100.80
```

Note: Header lines are lines of text that are not part of the data and do not share the same delimiter formatting as the data. Subheader lines also are not part of the data, but share the same delimiter formatting and therefore correspond to particular columns of data.

To Import this file

1. Select **File: Import: Single ASCII** from the menu to open the File Import dialog. Browse to the *\Samples\Import and Export* subfolder of the Origin program folder. Highlight the file *S15-125-03.dat*.

2. If you double-click this file or click the **Open** button, Origin will import the file automatically. To view the settings for how Origin will import the file, check the **Show Options Dialog** checkbox at the bottom of the dialog and then click **Open**. This will bring up the *impASC* X-Function dialog.



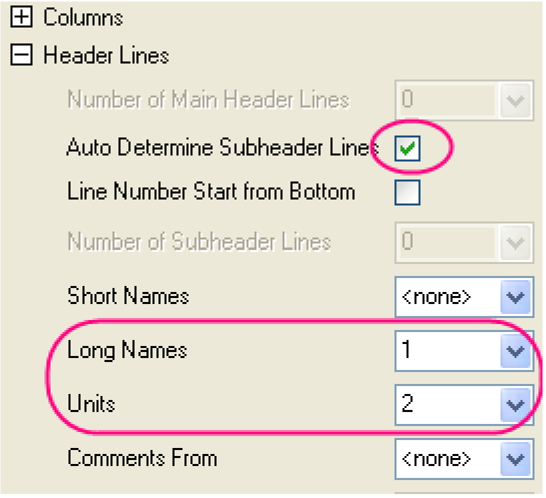
File name: S15-125-03.dat

Files of type: *.dat

☐ Open as read-only

☑ Show Options Dialog: <sheet>

3. Expand the **Import Options: Header Lines** tree node.



⊞ Columns

⊟ Header Lines

Number of Main Header Lines 0

Auto Determine Subheader Lines ☑

Line Number Start from Bottom ☐

Number of Subheader Lines 0

Short Names <none>

Long Names 1

Units 2

Comments From <none>

By default, Origin auto detects the subheader, and data will be imported from subheader. In this example, Origin automatically set the first line,

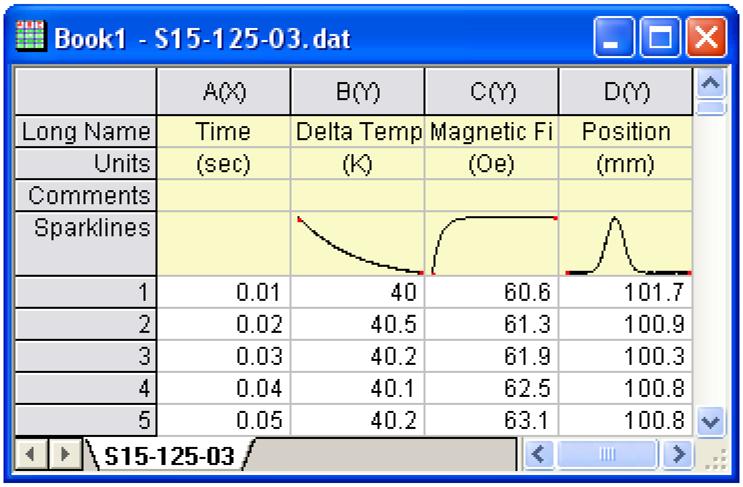
Time Delta Temperature Magnetic Field Position

as the worksheet column Long Name and the second line,

(sec) (K) (Oe) (mm)

as Units.

4. Click **OK** to accept these settings and import data into worksheet.



	A(X)	B(Y)	C(Y)	D(Y)
Long Name	Time	Delta Temp	Magnetic Fi	Position
Units	(sec)	(K)	(Oe)	(mm)
Comments				
Sparklines				
1	0.01	40	60.6	101.7
2	0.02	40.5	61.3	100.9
3	0.03	40.2	61.9	100.3
4	0.04	40.1	62.5	100.8
5	0.05	40.2	63.1	100.8

3.2 Import Wizard

3.2.1 Summary

The **Import Wizard** allows you to preview your file to help with importing of more complicated ASCII files. This is useful for files with many lines of header where you may wish to extract variables from both the import file name and the file header to later use for annotation on your graph.

Minimum Origin Version Required: Origin 8.0 SR6

3.2.2 What you will learn

This tutorial will show you how to use the **Import Wizard**.

3.2.3 Steps

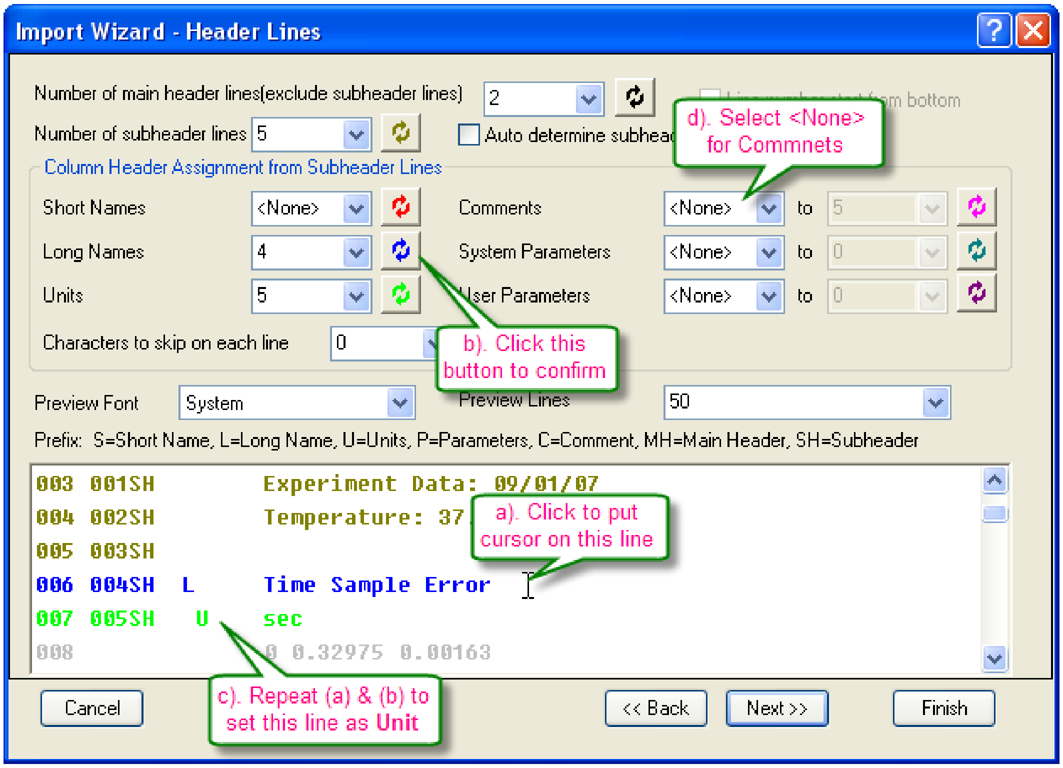
The **import wizard** allows you to import complicated ASCII files, extract variables from the import file name and header (for reuse in Origin), specify custom delimiters and date formats, or handle post-processing of your imported data using a custom LabTalk script. In addition, another chief advantage of using the Wizard to import your data files is that you can save your custom settings to a filter that can be used repeatedly to import the same or similarly-structured files.

1. Create a new workbook. Select **File: Import: Import Wizard** to open the *Import Wizard* dialog.

   Click the browser button … right beside **File** box. Browse to the *\Samples\Import and Export* folder and open the file *F1.dat*, and then click the **Next** button to navigate to **File Name Options** page.

2. Make sure the **worksheet with file name** box is checked to rename worksheet by imported file name. Click **Next** to navigate to **Header Lines** page.

3. This page enables you to easily customize the worksheet headers. For example, to specify the worksheet long name, put your cursor to the following line, and then click the button beside *Long Names*.



Number of main header lines(exclude subheader lines): 2

□ Line numbers start from bottom

Number of subheader lines: 5

□ Auto determine subheader

Column Header Assignment from Subheader Lines

Short Names	<None>	Comments	<None>	to 5
Long Names	4	System Parameters	<None>	to 0
Units	5	User Parameters	<None>	to 0

Characters to skip on each line: 0

Preview Font: System

Preview Lines: 50

Prefix: S=Short Name, L=Long Name, U=Units, P=Parameters, C=Comment, MH=Main Header, SH=Subheader

```
003  001SH        Experiment Data: 09/01/07
004  002SH        Temperature: 37
005  003SH
006  004SH  L     Time  Sample  Error
007  005SH   U    sec
008               0  0.32975  0.00163
```

d). Select <None> for Commnets

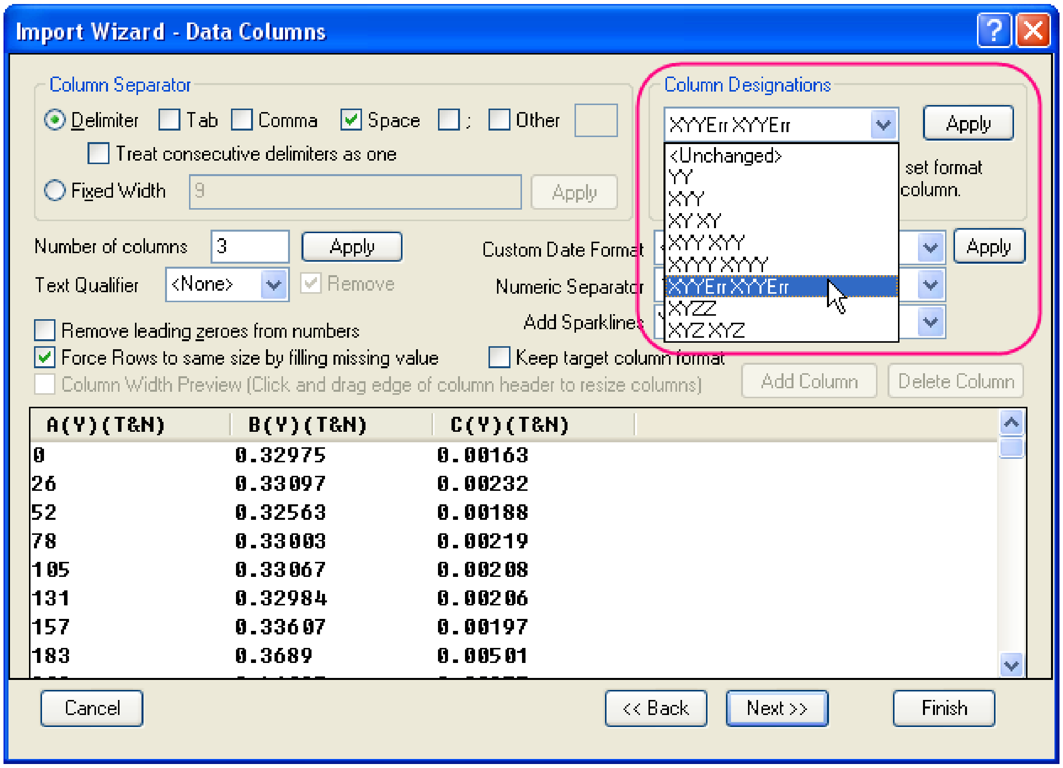
b). Click this button to confirm

a). Click to put cursor on this line

c). Repeat (a) & (b) to set this line as Unit

Similarly, specify the *Unit* line and select *<None>* for *Comments*.

4. Skip the next **Variable Extraction** page and go to the **Data Columns** page. Select **XYYErrXYYErr** from the *Column Designations* drop-down and click **Apply**.



Import Wizard - Data Columns

Column Separator

● Delimiter □ Tab □ Comma ☑ Space □ ; □ Other

□ Treat consecutive delimiters as one

○ Fixed Width 9

Number of columns 3

Text Qualifier <None> ☑ Remove

□ Remove leading zeroes from numbers

☑ Force Rows to same size by filling missing value

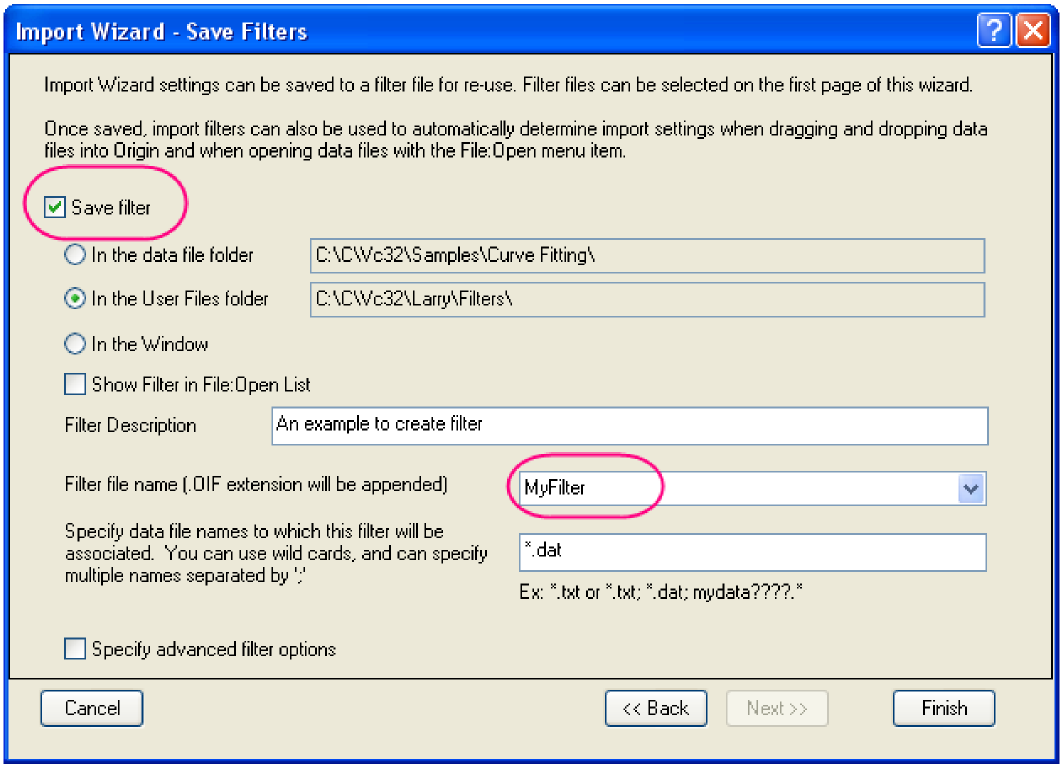
□ Keep target column format

□ Column Width Preview (Click and drag edge of column header to resize columns)

Column Designations XYYErr XYYErr

A(Y)(T&N)	B(Y)(T&N)	C(Y)(T&N)
0	0.32975	0.00163
26	0.33097	0.00232
52	0.32563	0.00188
78	0.33003	0.00219
105	0.33067	0.00208
131	0.32984	0.00206
157	0.33607	0.00197
183	0.3689	0.00501

5. Skip the **Data Selection** page, and go to the **Save Filters** page. To use these settings again, you can save this import procedure as a filter. Check the *Save filter* box and give a proper filter name in the *Filter file name* edit box (*MyFilter* in this example).



Import Wizard - Save Filters

Import Wizard settings can be saved to a filter file for re-use. Filter files can be selected on the first page of this wizard.

Once saved, import filters can also be used to automatically determine import settings when dragging and dropping data files into Origin and when opening data files with the File:Open menu item.

☑ Save filter

○ In the data file folder: C:\C\Vc32\Samples\Curve Fitting\

● In the User Files folder: C:\C\Vc32\Larry\Filters\

○ In the Window

□ Show Filter in File:Open List

Filter Description: An example to create filter

Filter file name (.OIF extension will be appended): MyFilter

Specify data file names to which this filter will be associated. You can use wild cards, and can specify multiple names separated by ';': *.dat

Ex: *.txt or *.txt; *.dat; mydata????.*

□ Specify advanced filter options

Click the **Finish** button to import the data.

3.3 Import Time Data

3.3.1 Summary

Origin interprets *Dates* based upon the Gregorian Calendar, while *Time* is interpreted in *hours*:*minutes*:*seconds*. When working with Date and Time data, Origin displays these data in different formats, but internally uses underlying numeric values for calculations and certainly plotting operations. This tutorial shows you how to import custom date/time data.

Minimum Origin Version Required: Origin 8.0 SR3

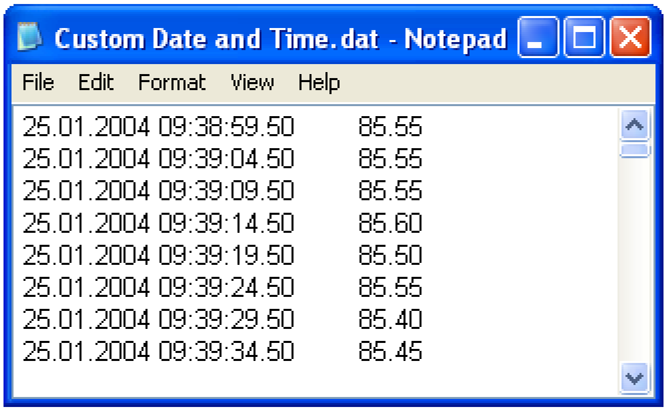
3.3.2 What you will learn

This tutorial covers:

- How to import data using multiple delimiters.
- How to define a custom date/time format
- How to change the display settings for the custom date/time format

3.3.3 Steps

1. We will import *\Samples\Import and Export\Custom Date and Time.dat* in this tutorial. Before importing the file, let's look at the data structure first.

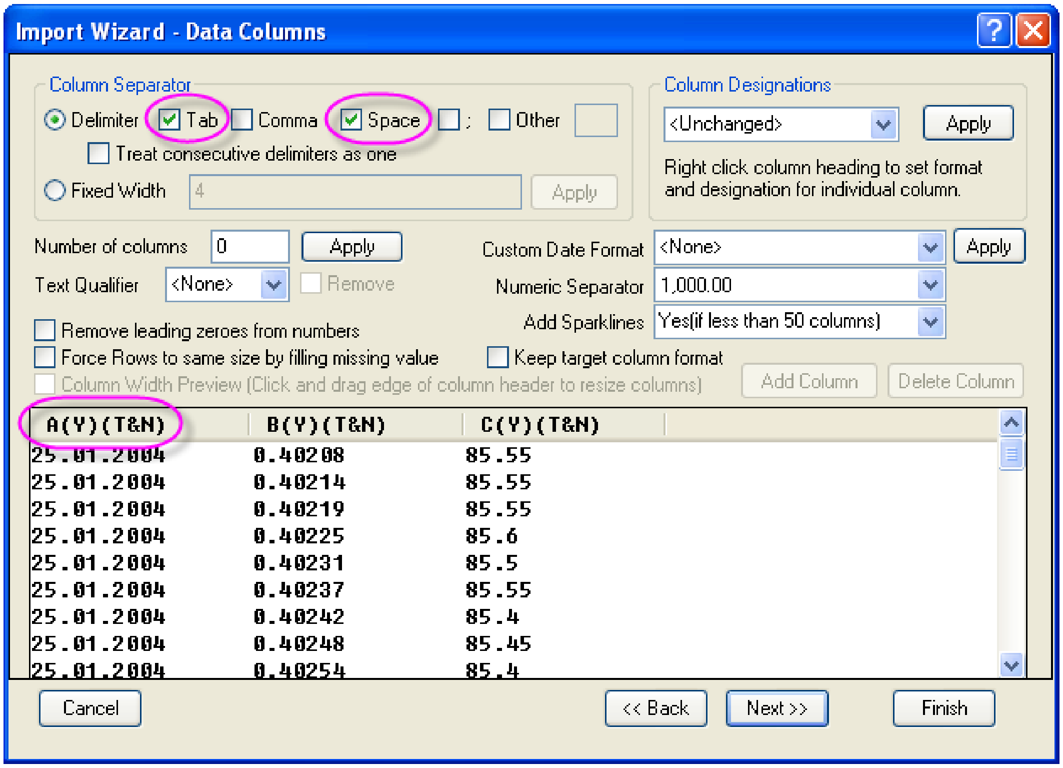


Custom Date and Time.dat - Notepad
File Edit Format View Help

```
25.01.2004 09:38:59.50	85.55
25.01.2004 09:39:04.50	85.55
25.01.2004 09:39:09.50	85.55
25.01.2004 09:39:14.50	85.60
25.01.2004 09:39:19.50	85.50
25.01.2004 09:39:24.50	85.55
25.01.2004 09:39:29.50	85.40
25.01.2004 09:39:34.50	85.45
```

We can see that there is a space between *Date* and *Time*, and it uses Tab to separate *Time* from the rest of the data. So we will use multiple delimiters to import this data file.

2. Open the file in **Import Wizard**. Accept the default settings on all pages until you get to the **Data Columns** page. Origin will, by default, use Tab to separate the data into two columns. To divide *Date* and *Time*, check the **Tab** and **Space** checkbox in the **Column Separator** group.



● Delimiter ☑ Tab ☐ Comma ☑ Space ☐ ; ☐ Other

☐ Treat consecutive delimiters as one

○ Fixed Width 4

<Unchanged>

Right click column heading to set format and designation for individual column.

Number of columns 0

Text Qualifier <None> ☐ Remove

Custom Date Format <None>

Numeric Separator 1,000.00

Add Sparklines Yes(if less than 50 columns)

☐ Remove leading zeroes from numbers

☐ Force Rows to same size by filling missing value

☐ Keep target column format

☐ Column Width Preview (Click and drag edge of column header to resize columns)

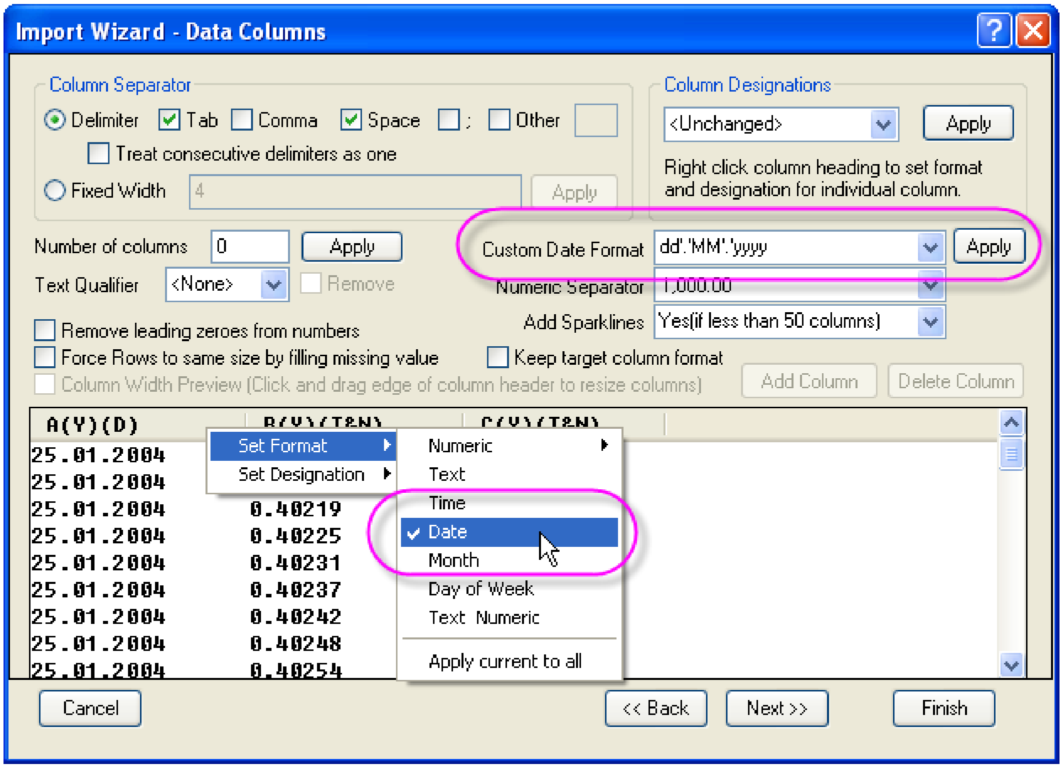
```
A(Y)(T&N)     B(Y)(T&N)     C(Y)(T&N)
25.01.2004    0.40208       85.55
25.01.2004    0.40214       85.55
25.01.2004    0.40219       85.55
25.01.2004    0.40225       85.6
25.01.2004    0.40231       85.5
25.01.2004    0.40237       85.55
25.01.2004    0.40242       85.4
25.01.2004    0.40248       85.45
25.01.2004    0.40254       85.4
```

Note in the preview box that the column title is **A(Y)(T&N)**, where **(T&N)** means the data format is *Text & Numeric*. Because the date uses "." to separate day, month and year, Origin by default treats the first column as *Text*. For the second *Time* column, Origin shows the underlying numeric values. To import data correctly, we should change the column properties.

3. In the **Custom Date Format** edit box, enter:

dd'.'MM'.'yyyy

where *dd*, *MM* and *yyyy* mean the days, months, and year respectively. Since the "."is used as a separator, we need to put single quotation marks around it in the format specification. After entering the custom format, press the **Apply** button next to the edit box. Then right-click on the header of the first column in the preview and select *Date* from the context menu:



Import Wizard - Data Columns

Column Separator

● Delimiter ☑ Tab □ Comma ☑ Space □ ; □ Other

□ Treat consecutive delimiters as one

○ Fixed Width 4 Apply

Column Designations

<Unchanged> Apply

Right click column heading to set format and designation for individual column.

Number of columns 0 Apply

Text Qualifier <None> □ Remove

Custom Date Format dd'.'MM'.'yyyy Apply

Numeric Separator 1,000.00

Add Sparklines Yes(if less than 50 columns)

□ Remove leading zeroes from numbers

□ Force Rows to same size by filling missing value

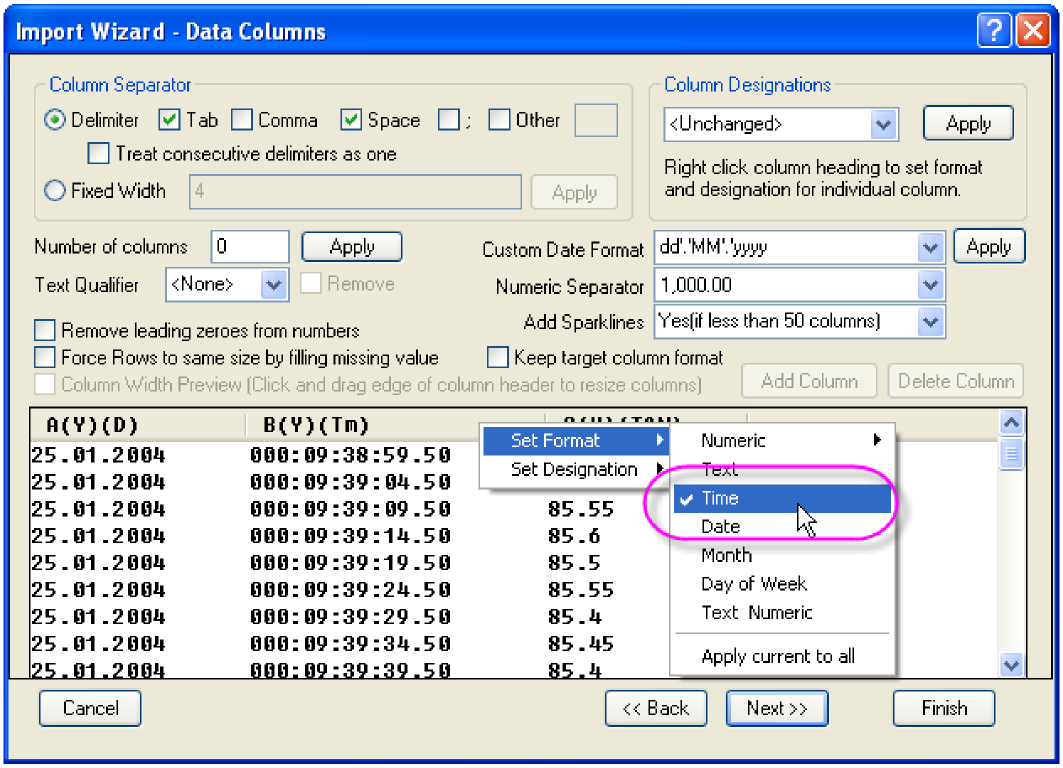
□ Keep target column format

□ Column Width Preview (Click and drag edge of column header to resize columns)

```
A(Y)(D)       B(Y)(T&N)     C(Y)(T&N)
25.01.2004
25.01.2004
25.01.2004    0.40219
25.01.2004    0.40225
25.01.2004    0.40231
25.01.2004    0.40237
25.01.2004    0.40242
25.01.2004    0.40248
25.01.2004    0.40254
```

Then the column title will turn into **A(Y)(D)** which means this is now a *Date* data column.

4. Similarly, right-click on the header of the second column and select *Time* to set that column as a time column:



● Delimiter ☑ Tab □ Comma ☑ Space □ ; □ Other

□ Treat consecutive delimiters as one

○ Fixed Width 4 Apply

<Unchanged> Apply

Number of columns 0 Apply

Custom Date Format dd'.'MM'.'yyyy Apply

Text Qualifier <None> □ Remove

Numeric Separator 1,000.00

Add Sparklines Yes(if less than 50 columns)

□ Remove leading zeroes from numbers

□ Force Rows to same size by filling missing value

□ Keep target column format

□ Column Width Preview (Click and drag edge of column header to resize columns)

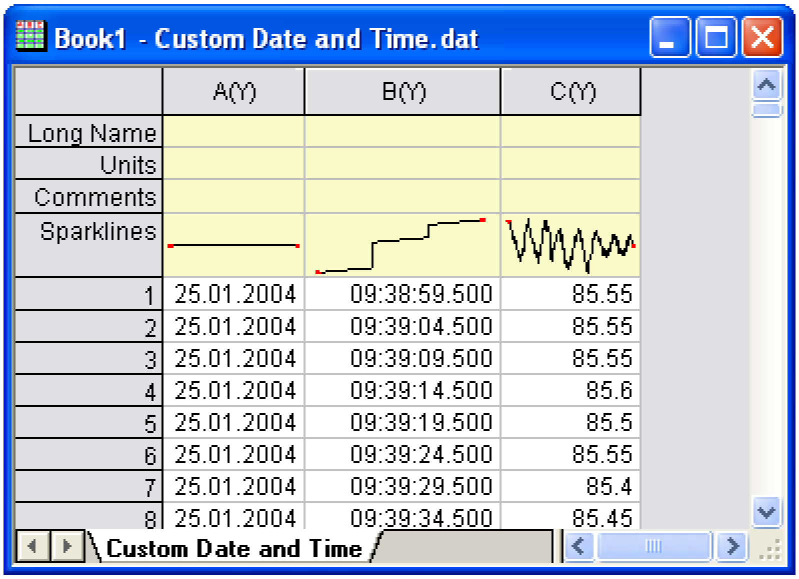
```
A(Y)(D)         B(Y)(Tm)
25.01.2004      000:09:38:59.50
25.01.2004      000:09:39:04.50
25.01.2004      000:09:39:09.50      85.55
25.01.2004      000:09:39:14.50      85.6
25.01.2004      000:09:39:19.50      85.5
25.01.2004      000:09:39:24.50      85.55
25.01.2004      000:09:39:29.50      85.4
25.01.2004      000:09:39:34.50      85.45
25.01.2004      000:09:39:39.50      85.4
```

You can see the time data display in long format. We can change the display setting after imported.

5. Click **Finish** button to import data. Then double-click the second column title to open the **Column Properties** dialog, and set the Time display as:

HH:mm:ss.##

The final worksheet data after imported will looks like:



	A(Y)	B(Y)	C(Y)
Long Name			
Units			
Comments			
Sparklines			
1	25.01.2004	09:38:59.500	85.55
2	25.01.2004	09:39:04.500	85.55
3	25.01.2004	09:39:09.500	85.55
4	25.01.2004	09:39:14.500	85.6
5	25.01.2004	09:39:19.500	85.5
6	25.01.2004	09:39:24.500	85.55
7	25.01.2004	09:39:29.500	85.4
8	25.01.2004	09:39:34.500	85.45

Note: In the case of this particular data file, the first column simply has the exact same date in every row. So at this point you may want to set this first column as Disregard by right-clicking on the column header and selecting *Disregard* from the context menu. Then you can set the 2nd column as type X, and plot the data in the third column against the time data in the 2nd column.

3.4 Post Processing with Import Filter

3.4.1 Summary

The Import Wizard allows defining a custom filter to import ASCII and simple binary files. The filter can then be reused with similar data files once created. The filter mechanism also allows including LabTalk script lines that will then be run at the end of the import. This capability allows user to add post-processing script code to the filter.

Minimum Origin Version Required: Origin 8.0 SR6

3.4.2 What you will learn

- How to add post processing script to existing import wizard filter

3.4.3 Steps

1. Start a new workbook by clicking the **New Workbook** button on the Standard toolbar.
2. Click the **Import Wizard** button on the Standard toolbar to open the wizard.
3. In the **Data Source** group, click the button to the right of **File** and navigate to and add the file **Samples\Import and Export\S15-125-03.dat**.
4. A suitable filter for importing this file already exists in the data folder. Click the **Next** button to proceed thru all the pages of the wizard till you get to the **Save Filters** page.
5. On this page, check the **Save Filter** check box and also the **Specify advanced filter options** check box and then click **Next**. This will bring you to the **Advanced Options** page.
6. On this page, copy and paste the following lines in the edit box:

```
nlbegin iy:=(1,4) func:=gauss;
nlfit;
nlend output:=1 autoupdate:=au_auto;
```

7. Press **Finish**. This will save the filter along with these added lines of script, and the file will be imported and the script will run. The workbook will then have three sheets, which will include the custom report sheet and the fitted curve sheet, which are results of gaussian function fit to column 4 of the imported data.
8. Start a new project and open import wizard again and add all three files **S15_125_03.dat, S21-235-07, S32-014-04** in the file open dialog.
9. Check to see that the **Import mode** drop-down is set to **Start New Books** and click Finish. Your modified filter will be used and after each file is imported, the 4th column will be fit with the gaussian function.

4 Data Manipulation

- Extract Worksheet Data

4.1 Extract Worksheet Data

4.1.1 Summary

This tutorial will show you how to use the **Extract Worksheet Data** dialog.

Minimum Origin Version Required: Origin 8.0 SR6

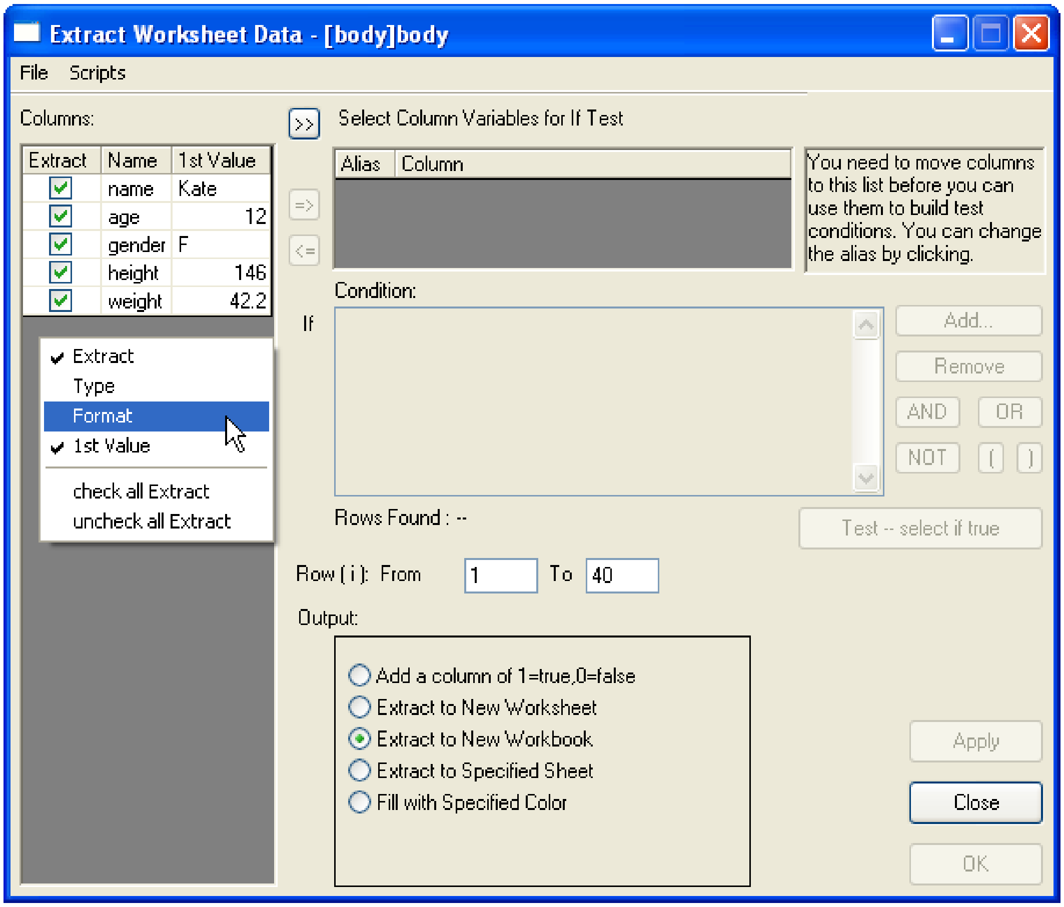
4.1.2 What you will learn

This tutorial will show you how to:

- Extract numeric and time data
- Use an alias in an extraction condition
- Use LabTalk functions in an extraction condition

4.1.3 Dialog overview

Start with a new workbook and import the file \Samples\Statistics\body.dat, click the menu item **Worksheet: Extract Worksheet Data** to open the dialog as follows:



Extract Worksheet Data - [body]body

Columns:

Extract	Name	1st Value
☑	name	Kate
☑	age	12
☑	gender	F
☑	height	146
☑	weight	42.2

Select Column Variables for If Test

Alias	Column

You need to move columns to this list before you can use them to build test conditions. You can change the alias by clicking.

Condition:

Rows Found : --

Row (i): From 1 To 40

Output:

○ Add a column of 1=true,0=false
○ Extract to New Worksheet
● Extract to New Workbook
○ Extract to Specified Sheet
○ Fill with Specified Color

Basically, there are two main panels in the **Extract Worksheet Data** dialog. The left panel lists all the columns in the active worksheet, you can right-click and select some column properties you want to see, such as *Format*, *1st Value*, etc.

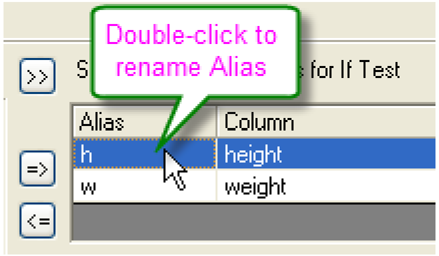
Note the **Extract** column in this panel, only data selected in the Extract checkbox will be extracted.

The right panel is where you set and test extract conditions. For example, you can select the column you want to use in the extract condition, and then click the => button to move it into the **Select Column Variable for If Test** group.

4.1.4 Set the conditions

Extract Numerical Data

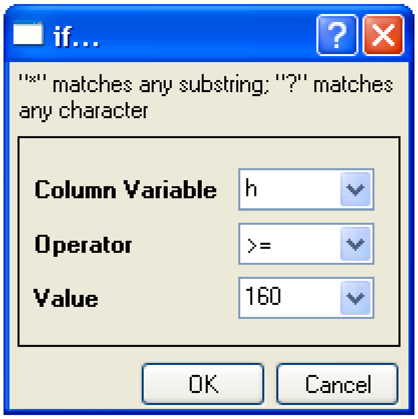
When there are available columns in the **Select Column Variable for If Test** group, the **Condition** edit box becomes editable for you to set conditions. For example, select *height* and *weight* to the group, Origin will automatically set an alias for each column. You can click into the **Alias** cell and rename the alias:



Alias	Column
h	height
w	weight

These alias can be used directly in the extract condition. Let's keep the default alias, *h* and *w* in this example.

The buttons on the right side of **Condition** edit box can be help to establish extract conditions. For example, to extract data that height is greater and equal to 160cm, highlight the column on **Select Column Variable for If Test** and click **Add** and build the first condition as follow:



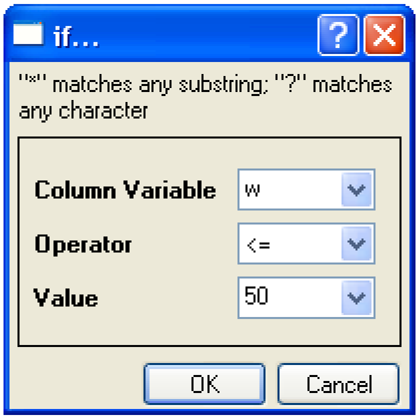
"*" matches any substring; "?" matches any character

Column Variable: h

Operator: >=

Value: 160

When there are multiple conditions, you can also combine these conditions by logical operation. Click **AND** button to add one more condition. Then highlight w on **Select Column Variable for If Test** group and click **ADD** again, this time, we are looking for weight less than or equal to 50kg:



"*" matches any substring; "?" matches any character

Column Variable: w

Operator: <=

Value: 50

When the condition is done, click the **Test -- select if true** button and Origin will return 5 found records. Of course, if you familiar to logical operation syntax, you can type the condition on the edit box directly:

```
h>=160 AND w<=50
```

Accept other default settings and click the **OK** button. A new workbook is created with these 5 records.

Extract Strings

When extracting strings, you need to enclose the string by double quotation marks ". For example, select the *gender* column into the **Select Column Variables for If Test** group. Using the alias *g*, you can extract all female data by:

```
g == "F"
```

Extract Time Data

Date and Time data are internally saved as numeric values in Origin. Date is the integer part of the numeric value, while Time is the fractional part. In Origin, you can use the *int()* and *frac()* functions to return the integer and fractional part of a number, and use the *Date(MM/DD/YY)* and *Time(HH:mm:ss)* functions to transfer string to time data. We can combine these functions to extract time data.

For example, using data from Import Time Data tutorial, you can extract data within time period 10:00 ~ 11:00 by:

```
frac(B) > Time(10:00:00) AND frac(B) < Time(11:00:00)
```

You can see Origin found 120 records. Similarly, if you want to extract *Date* data, you can try some condition like:

```
int(A) > Date(01/24/2004)
```

5 Graphing

- Basic 2D Plotting
- Basic 3D Plotting
- Customizing Graphs
- Adding a Data Plot to Existing Graph
- Add multiple fitted curves in a Histogram
- Copy and Apply a Plot Format to Another Plot
- Adding and Arranging Layers
- Create an 8 layer multi-panel plot template
- Mark out a segment of plot with different plot style
- Simple Dot Chart
- Multi-Data Dot Chart
- Plot Functions with Parameters Defined in a Worksheet

5.1 Basic 2D Plotting

5.1.1 Summary

Origin provides flexible ways to create 2D plots. You can easily customize plot attributes, arrange layers, and select different datasets for each layer. This tutorial will teach you the basic plotting skills.

Minimum Origin Version Required: Origin 8.0 SR6

5.1.2 What you will learn

This tutorial will show you how to:

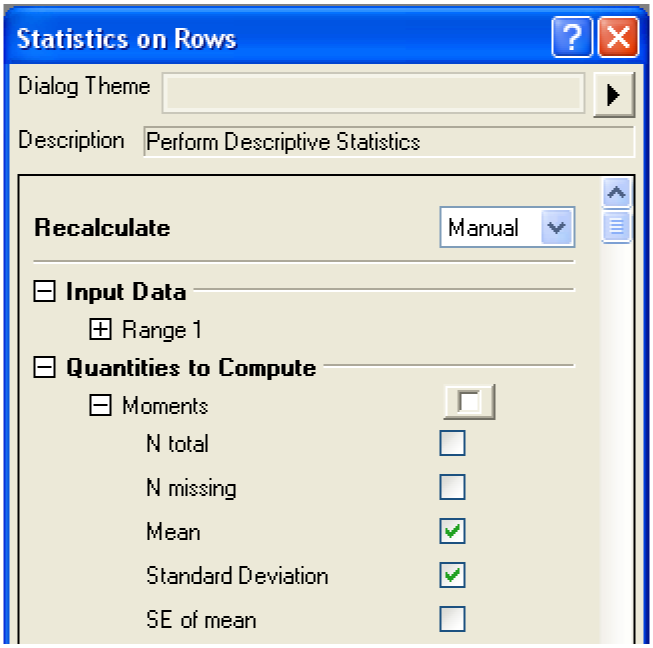
- Perform simple row statistics
- Create a graph and save as a template
- Plot into your template
- Use the Plot Setup dialog

5.1.3 Steps

Simple row statistics

1. Start with an empty worksheet, select **File: Import: Single ASCII...** to open the *Import Single ASCII* dialog, browse to the *\Samples\Curve Fitting* subfolder of the Origin program folder, and import the file *Dose Response - No Inhibitor.dat*.

2. Highlight columns 2 through 4 and select **Statistics: Descriptive Statistics: Statistics on Rows**. Make sure to check the **Mean** and **Standard Deviation** check boxes on *Quantities to Compute* > *Moments* branch to output these results.



Statistics on Rows

Dialog Theme

Description: Perform Descriptive Statistics

Recalculate: Manual

⊟ **Input Data**

⊞ Range 1

⊟ **Quantities to Compute**

⊟ Moments □

N total □

N missing □

Mean ☑

Standard Deviation ☑

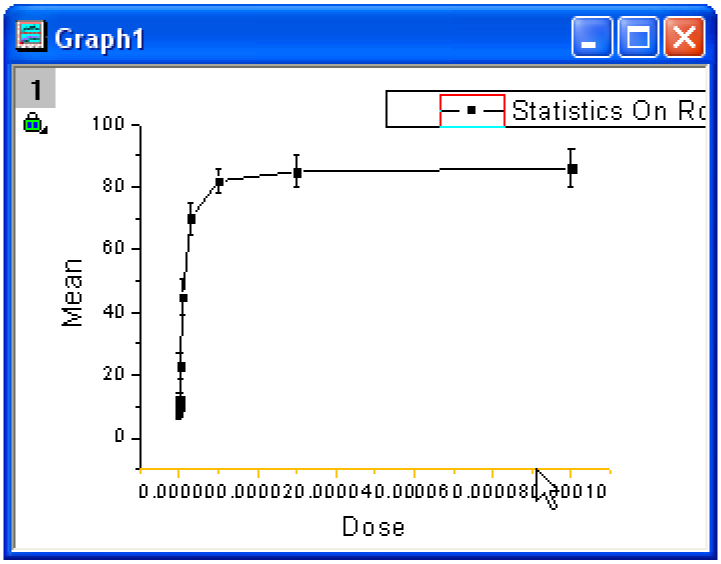
SE of mean □

3. After you click the **OK** button, two new columns, **Mean(Y)** and **SD(yErr)** are added to the source worksheet. Here, **yErr** means that this is an error column and the data in this column can be used to plot error bars.

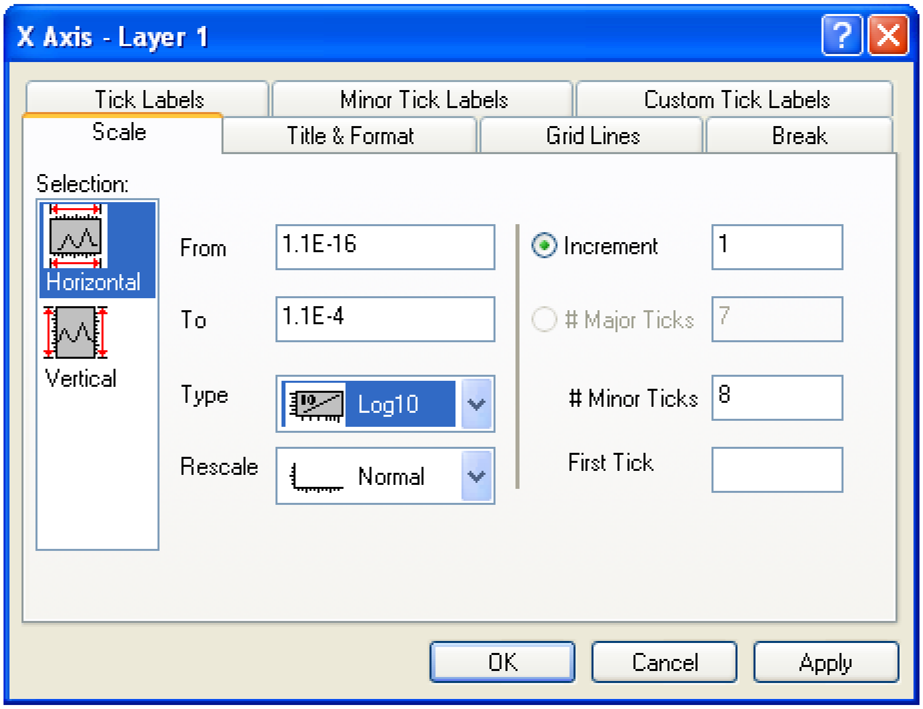
Note: To simplify plotting, each column in an Origin worksheet has a plotting designation. To change a column's plotting designation, select the column and click on the **Column** menu. Alternatively, right-click on the column and choose **Set As** from the context menu.

Create a graph and save as a template

1. Highlight the Mean(Y) and SD(yEr-) columns and select **Plot: Line+Symbol: Line+Symbol** to create the plot:



2. To change the X scale to Log, double-click on the X axis to bring up the **X Axis** dialog. On the **Scale** tab, change the axis **Type** to **Log10**:



From 1.1E-16

To 1.1E-4

Type Log10

Rescale Normal

● Increment 1

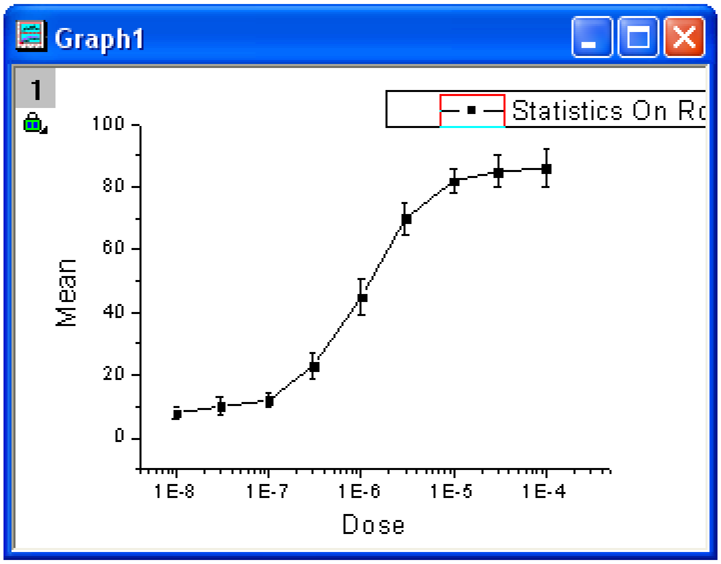
○ # Major Ticks 7

Minor Ticks 8

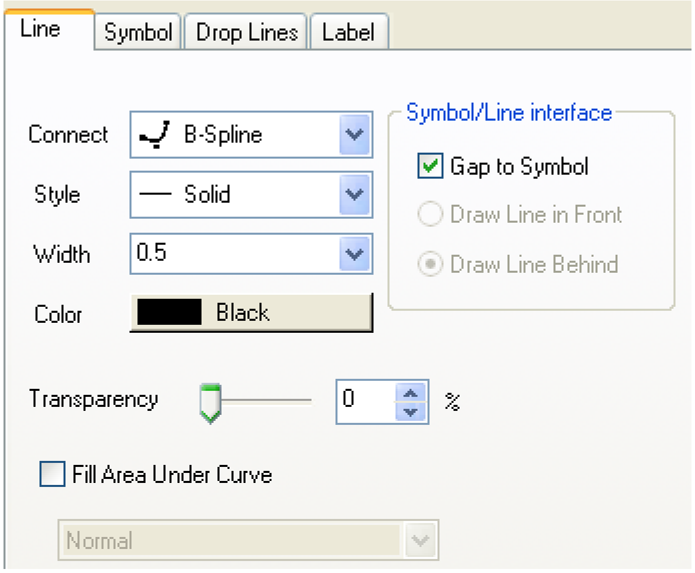
First Tick

Click the **OK** button to close the dialog.

3. Select **Graph: Rescale to Show All** from the menu, which will rescale the X and Y axes of the graph. :



4. To edit the curve, double-click on any plot symbol to bring up the **Plot Details** dialog. Alternatively, right-click inside the graph and choose **Plot Details** from the context menu. On the **Line** tab of right panel, select **B-Spline** as connect line to get a smoother curve.



Connect: B-Spline
Style: Solid
Width: 0.5
Color: Black

Symbol/Line interface
☑ Gap to Symbol
○ Draw Line in Front
● Draw Line Behind

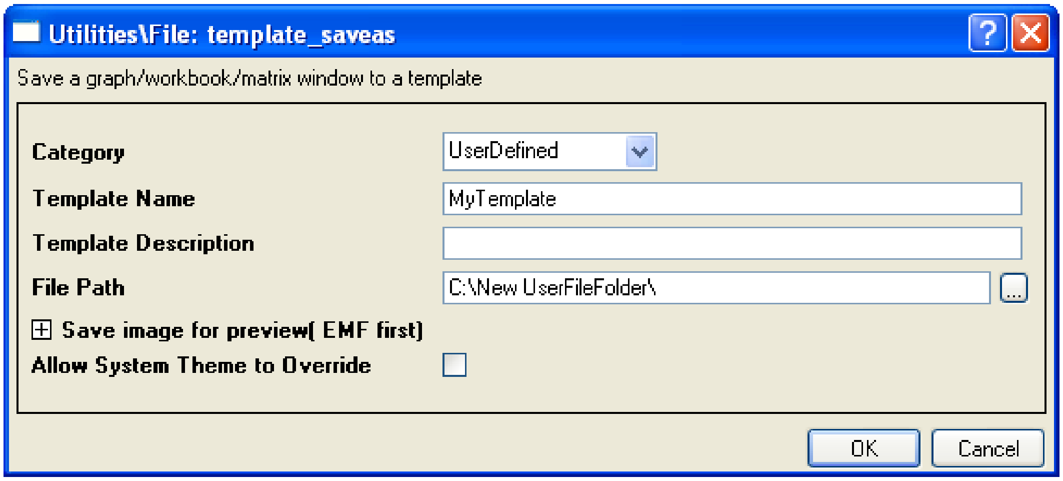
Transparency: 0 %

□ Fill Area Under Curve

Normal

Click the **OK** button to close the dialog.

5. When all modifications have been made and the graph looks the way you want it, you can use this graph to create a template, to be used in the future with similar data. Select **File: Save Template as** to open the *Save Template* dialog. In the **Category** drop-down list, select *UserDefined*; and then type a proper **Template Name**. In this example, we use *MyTemplate*. Click **OK** to save the template.



Utilities\File: template_saveas

Save a graph/workbook/matrix window to a template

Category: UserDefined

Template Name: MyTemplate

Template Description:

File Path: C:\New UserFileFolder\

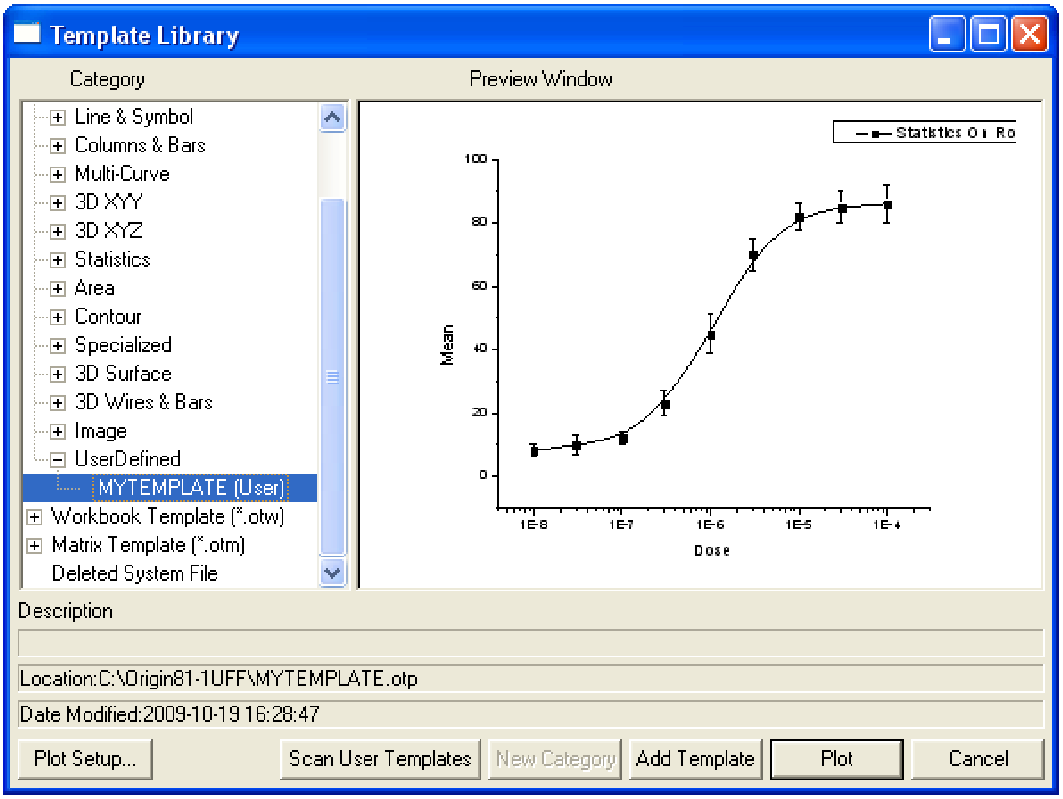
⊞ Save image for preview(EMF first)

Allow System Theme to Override □

Plot into graph template with the Plot Setup dialog

1. Click the ▦ button to open a new workbook, and import the file *Samples**Curve Fitting**Dose Response - Inhibitor.dat* as above. Perform **Statistics on Rows**, calculating the Mean and SD of this worksheet as you did above and by following the same steps.

2. Select **Plot: Template Library** to open the *Template Library*. Select *MyTemplate* from the *UserDefined* category.

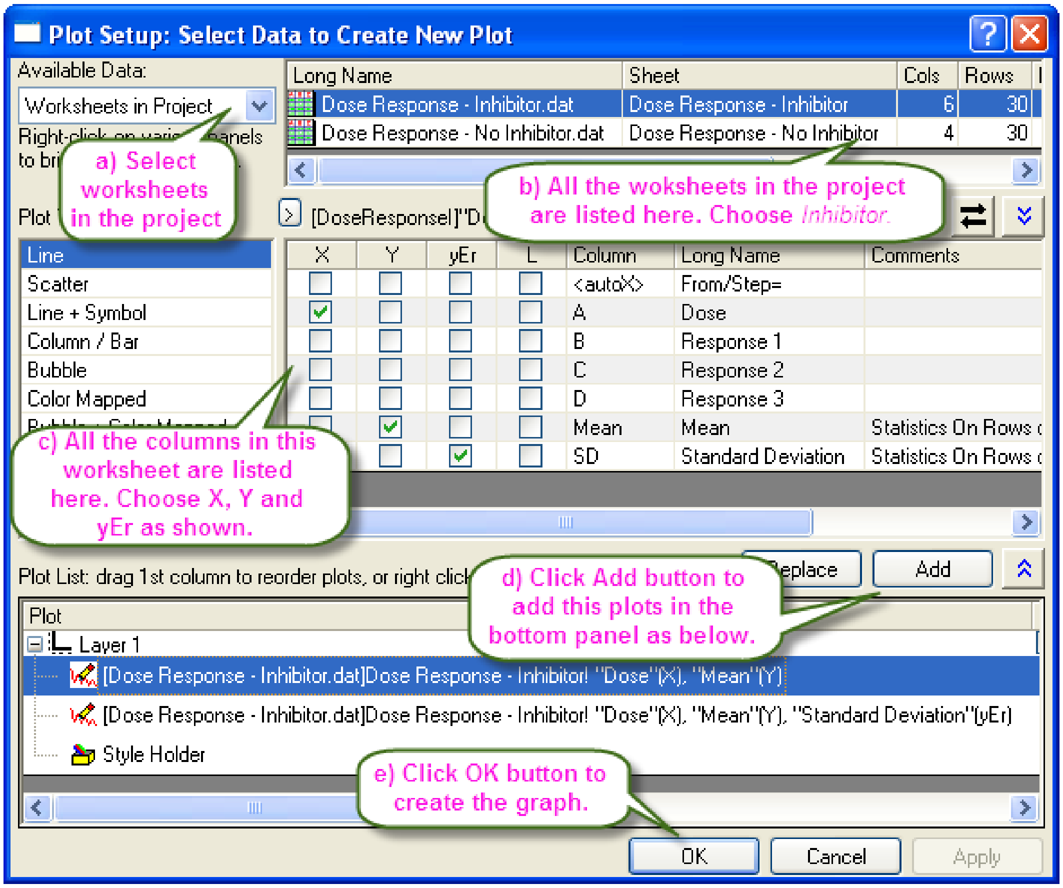


Location:C:\Origin81-1UFF\MYTEMPLATE.otp

Date Modified:2009-10-19 16:28:47

Click the **Plot Setup** button to select the data from which to create the plot. If you click the Plot button, Origin will plot whatever data is highlighted in the worksheet.

3. In the *Plot Setup* dialog, you can choose which columns are to be plotted. (There are three panels in *Plot Setup* dialog, click the ⏬ or ⏫ button to expand them) To finish creating the plot from your template, please follow the steps a - e outlined on the picture below.



Plot Setup: Select Data to Create New Plot

Available Data: Worksheets in Project

Long Name	Sheet	Cols	Rows
Dose Response - Inhibitor.dat	Dose Response - Inhibitor	6	30
Dose Response - No Inhibitor.dat	Dose Response - No Inhibitor	4	30

[DoseResponseI]"D

X	Y	yEr	L	Column	Long Name	Comments
□	□	□	□	<autoX>	From/Step=	
☑	□	□	□	A	Dose	
□	□	□	□	B	Response 1	
□	□	□	□	C	Response 2	
□	□	□	□	D	Response 3	
□	☑	□	□	Mean	Mean	Statistics On Rows (
	□	☑	□	SD	Standard Deviation	Statistics On Rows (

Plot List: drag 1st column to reorder plots, or right click

Plot
- Layer 1
  - [Dose Response - Inhibitor.dat]Dose Response - Inhibitor! "Dose"(X), "Mean"(Y)
  - [Dose Response - Inhibitor.dat]Dose Response - Inhibitor! "Dose"(X), "Mean"(Y), "Standard Deviation"(yEr)
  - Style Holder

a) Select worksheets in the project

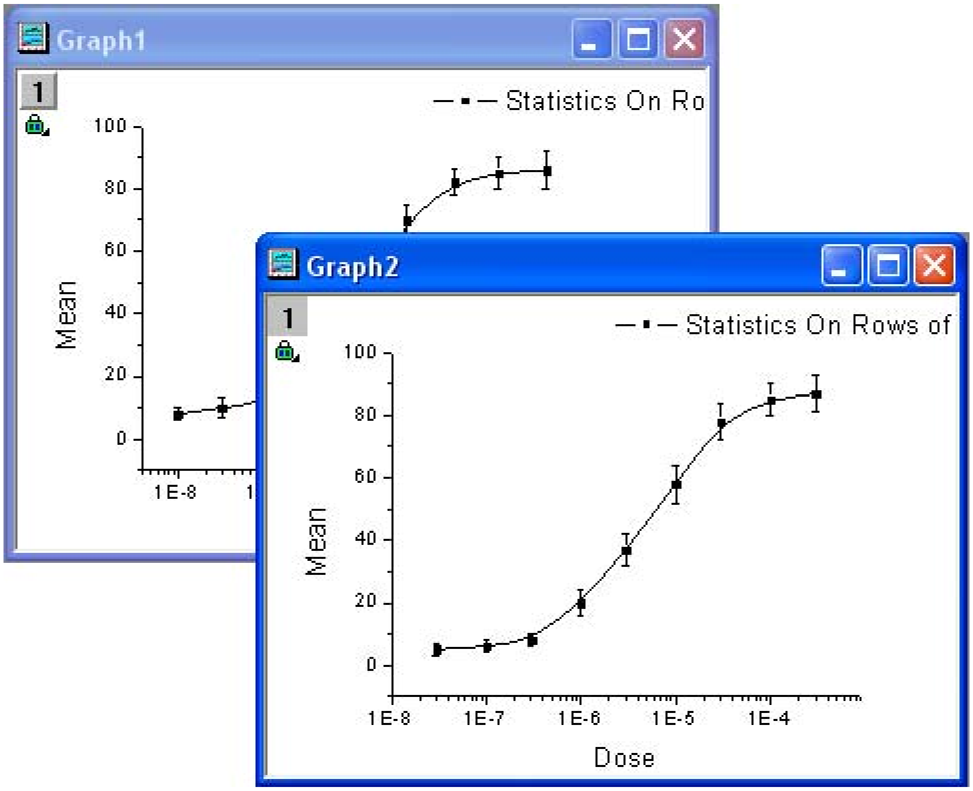
b) All the woksheets in the project are listed here. Choose *Inhibitor*.

c) All the columns in this worksheet are listed here. Choose X, Y and yEr as shown.

d) Click Add button to add this plots in the bottom panel as below.

e) Click OK button to create the graph.

And then you will have:



5.2 Basic 3D Plotting

5.2.1 Summary

In Origin, most 3D plots -- including 3D surface, wire frame/wire surface, 3D bar plot and 2D contour -- are created from an Origin matrix. In most cases, the raw data is XYZ data and you should convert it to a matrix first, using one of Origin's built-in gridding routines.

Minimum Origin Version Required: Origin 8.0 SR6

5.2.2 What you will learn

This tutorial will show you how to:

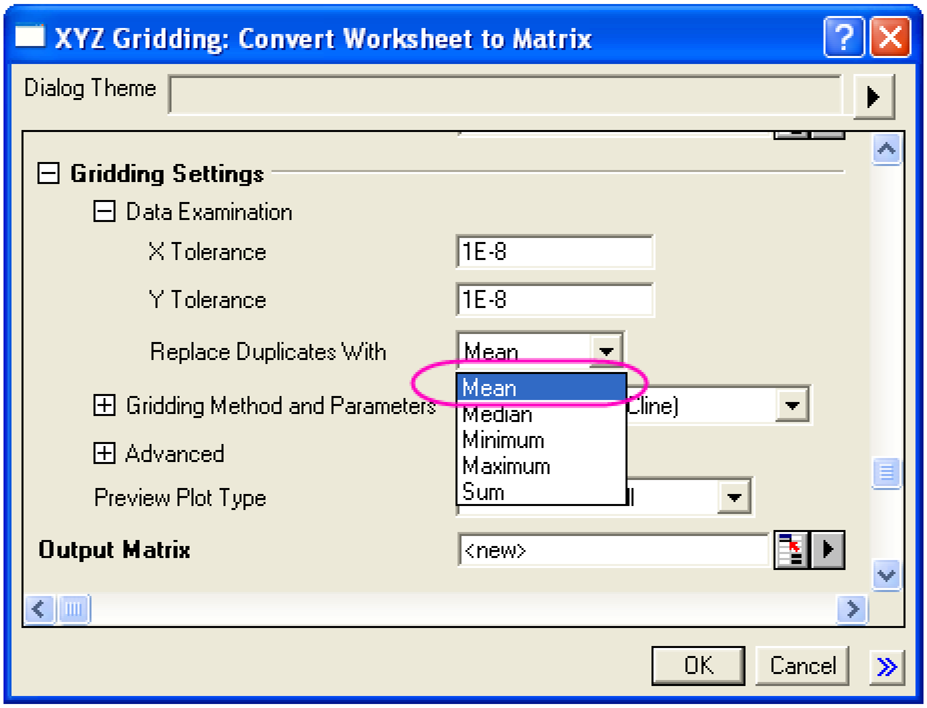
- Create a 3D graph in Origin
- Convert Worksheet data to a Matrix
- Use the layer contents dialog to add/remove dataset
- Use the Plot Details dialog to modify graph

5.2.3 Steps

1. Import the file *\Samples\Matrix Conversion and Gridding\XYZ Random Gaussian.dat*.

2. Highlight the 3rd column, right-click, and select **Set As: Z** from the context fly-out menu.

3. To convert the worksheet XYZ data into a matrix, select **Worksheet: Convert to Matrix: XYZ Gridding** to bring up the **XYZ Gridding** dialog. Select **Mean** from the drop-down list of the **Replace Duplicates with** item as below:



XYZ Gridding: Convert Worksheet to Matrix

Dialog Theme

Gridding Settings

Data Examination

X Tolerance 1E-8

Y Tolerance 1E-8

Replace Duplicates With Mean

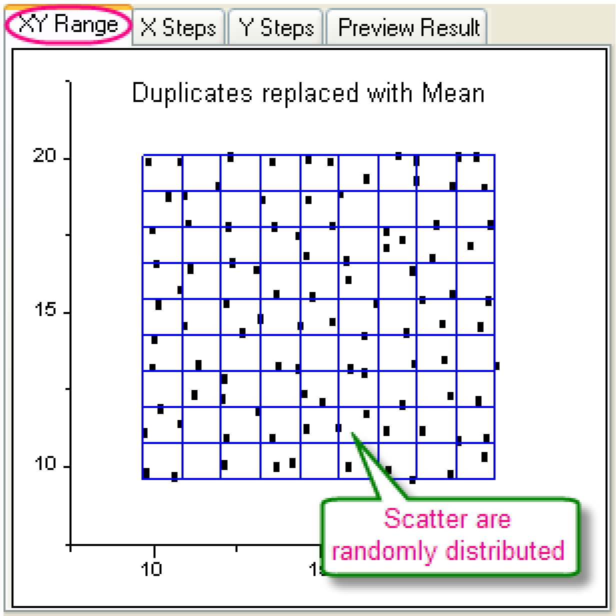
Gridding Method and Parameters

Advanced

Preview Plot Type

Output Matrix <new>

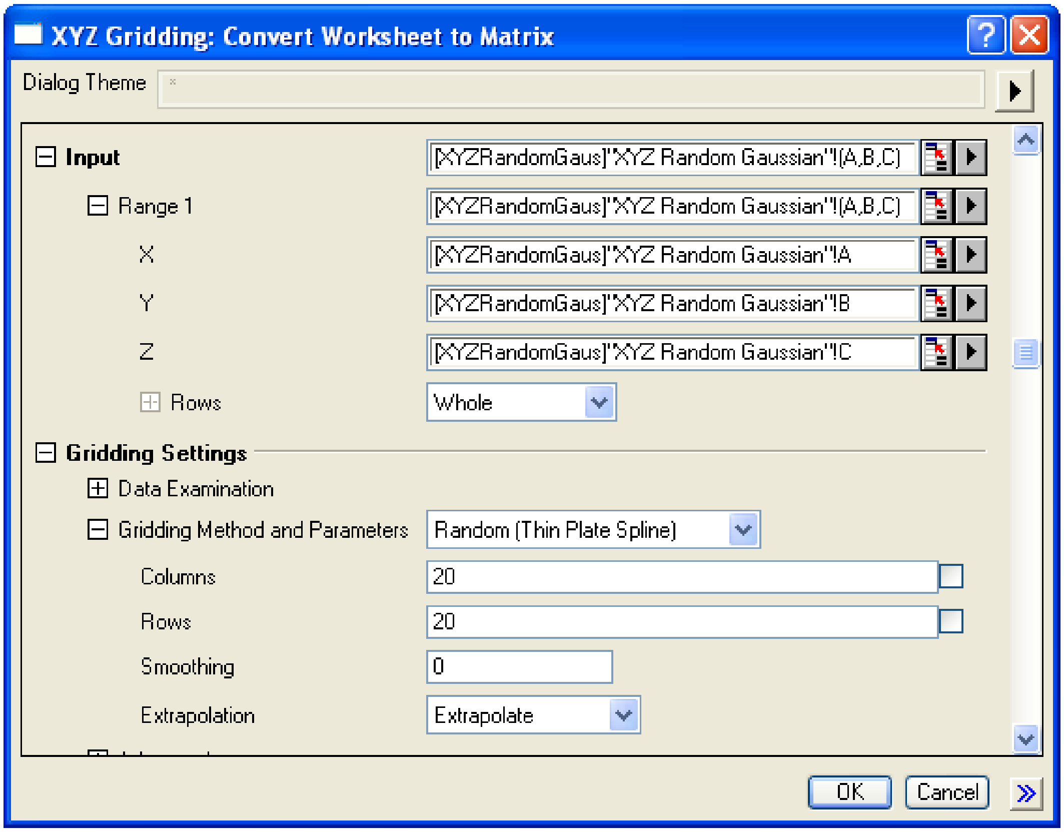
4. After that, you can see the right preview panel as below.As you can see, the XY data are randomly distributed, so a random gridding method should be used.



Duplicates replaced with Mean

Scatter are randomly distributed

5. Use the following settings and click **OK** to convert the XYZ columns of data into a matrix of data. The TPS gridding method will generate a smooth surface.

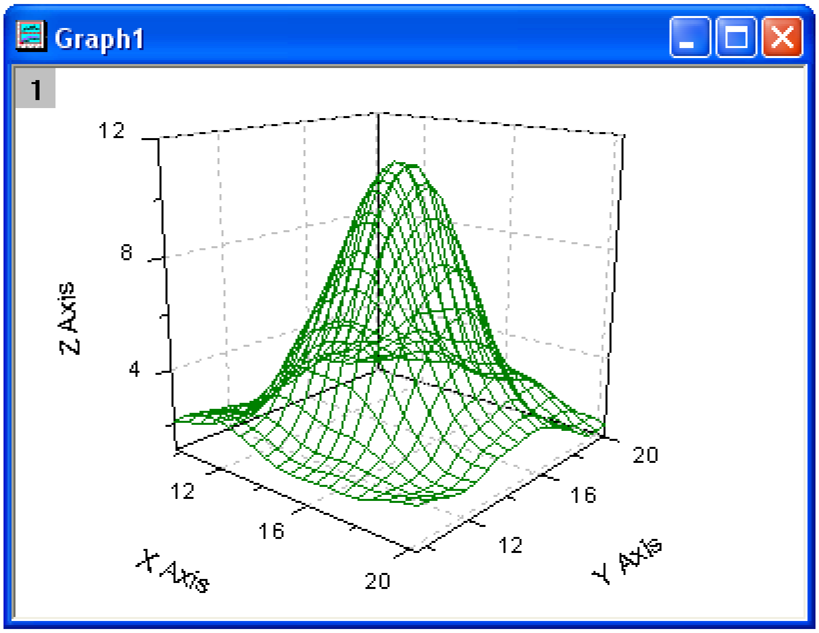


XYZ Gridding: Convert Worksheet to Matrix

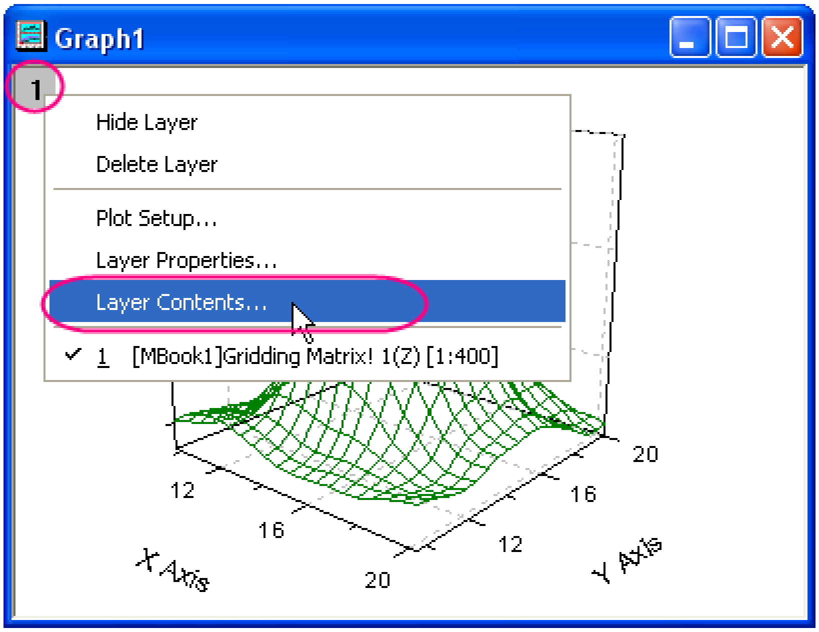
Dialog Theme *

- ⊟ Input: [XYZRandomGaus]"XYZ Random Gaussian"!(A,B,C)
  - ⊟ Range 1: [XYZRandomGaus]"XYZ Random Gaussian"!(A,B,C)
    - X: [XYZRandomGaus]"XYZ Random Gaussian"!A
    - Y: [XYZRandomGaus]"XYZ Random Gaussian"!B
    - Z: [XYZRandomGaus]"XYZ Random Gaussian"!C
    - ⊞ Rows: Whole
- ⊟ Gridding Settings
  - ⊞ Data Examination
  - ⊟ Gridding Method and Parameters: Random (Thin Plate Spline)
    - Columns: 20 □
    - Rows: 20 □
    - Smoothing: 0
    - Extrapolation: Extrapolate

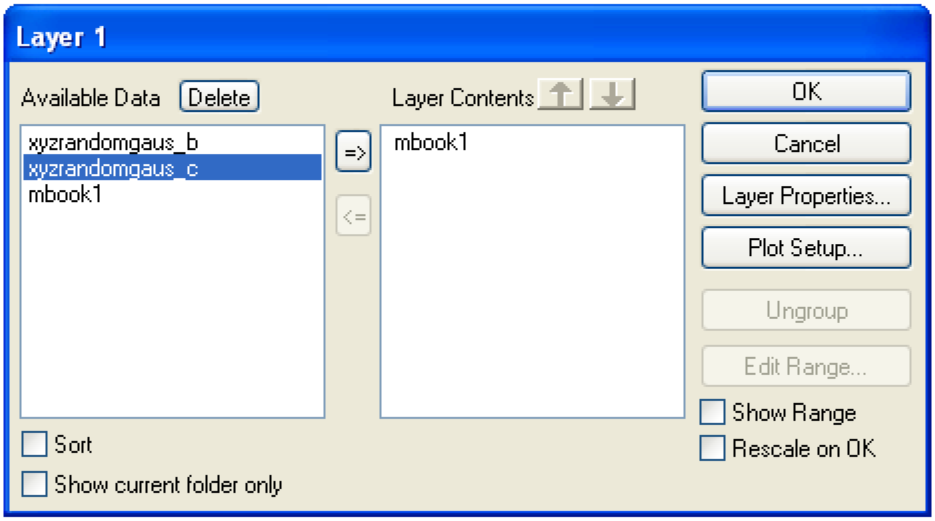
6. Set the new generated matrix as the active window and select **Plot: 3D Wires and Bars: Wire Frame** from the menu to plot a 3D mesh:



7. In order to plot the original data points on the graph, you can use the **Layer Contents**. Right-click on the layer icon and select **Layer Contents**.



In the Layer Contents dialog, select the worksheet Z column (In this example, xyzrandomgaus_c) and add it into the Layer Contents list.



Available Data: xyzrandomgaus_b, xyzrandomgaus_c, mbook1

Layer Contents: mbook1

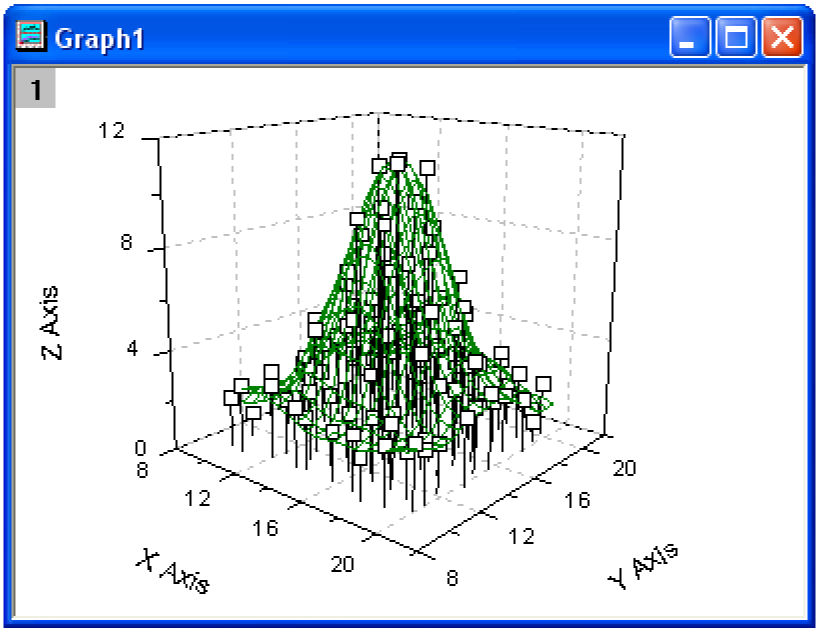
□ Sort

□ Show current folder only

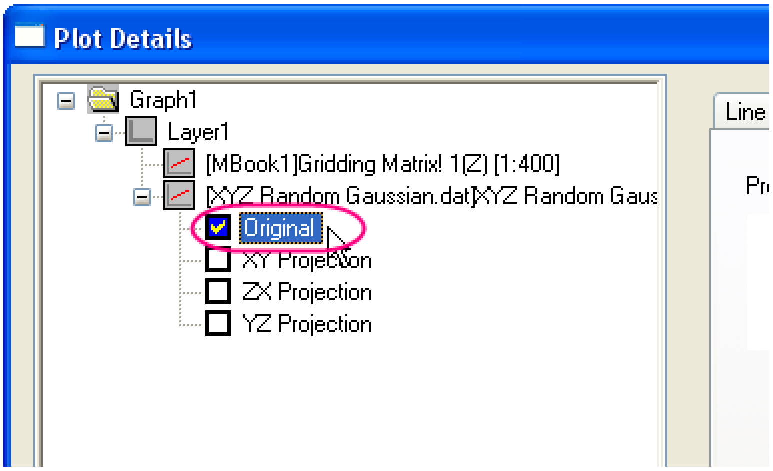
□ Show Range

□ Rescale on OK

After you click **OK**, the source data points will be added to the layer.

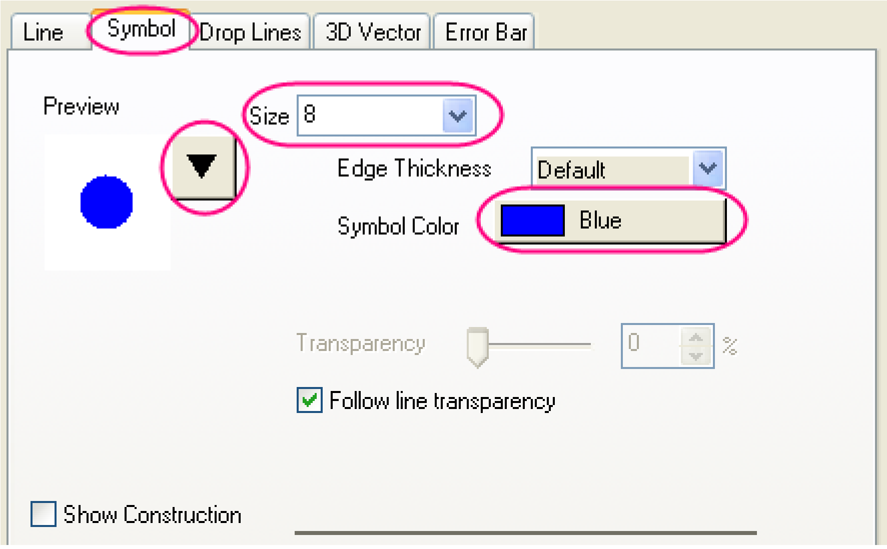

Graph1
1
12
8
4
0
Z Axis
8
12
16
20
X Axis
8
12
16
20
Y Axis

8. You can now use the **Plot Details** dialog to modify the appearance. Double-click on the graph to bring up the **Plot Details** dialog. On the left panel, select the 3D scatter data:



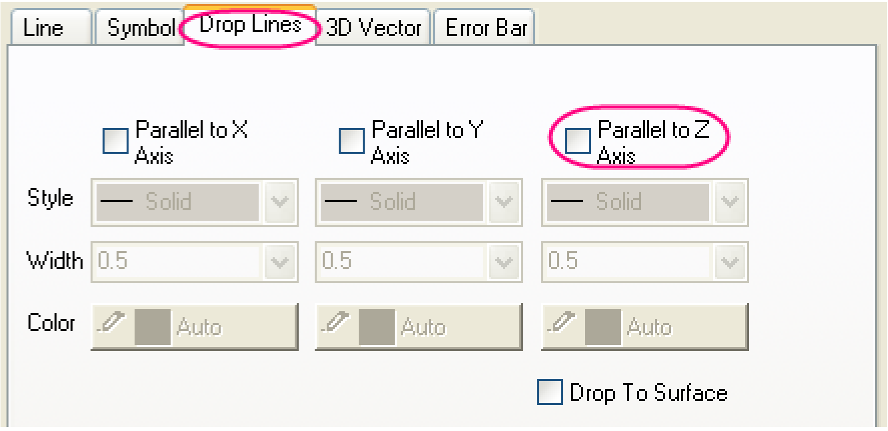
Graph1
Layer1
[MBook1]Gridding Matrix! 1(Z) [1:400]
[XYZ Random Gaussian.dat]XYZ Random Gaus
☑ Original
□ XY Projection
□ ZX Projection
□ YZ Projection

And then go to the **Symbol** tab on the right panel, adjust the symbol type, size and color, etc.



Size 8
Edge Thickness Default
Symbol Color Blue
Transparency 0 %
☑ Follow line transparency
□ Show Construction

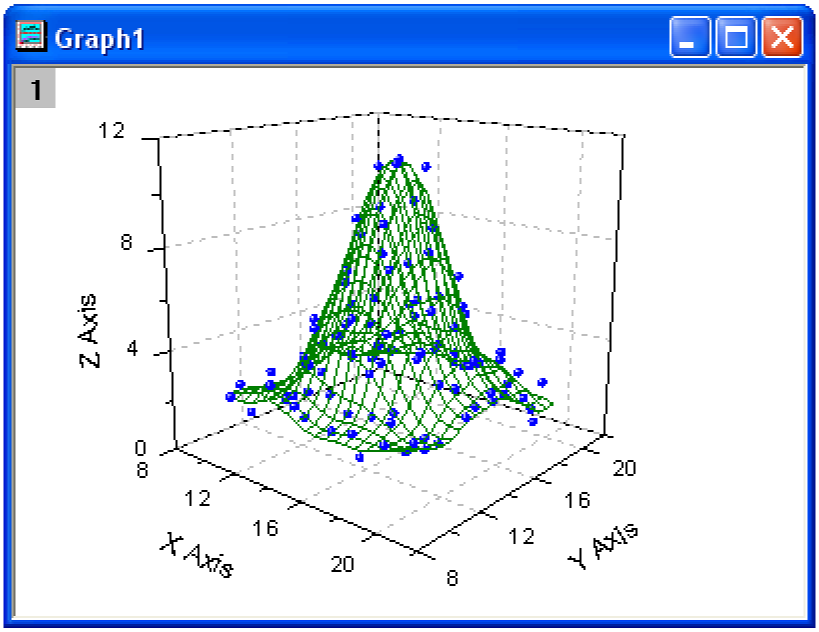
Remove the drop lines on the **Drop Lines** panel:



	□ Parallel to X Axis	□ Parallel to Y Axis	□ Parallel to Z Axis
Style	Solid	Solid	Solid
Width	0.5	0.5	0.5
Color	Auto	Auto	Auto

□ Drop To Surface

9. When done, click **OK** to accept the modifications:



5.3 Adding a Data Plot to Existing Graph

5.3.1 Summary

The Plot Setup dialog can be used to add/reorder/arrange data plots in an exiting graph. This dialog provides flexibility in selecting the desired data sheet and then selecting data to be plotted using column meta data such as long name.

Minimum Origin Version Required: Origin 8.0 SR6

5.3.2 What you will learn

This tutorial will show you how to:

- Use the Plot Setup Dialog Top Panel to find your dataset

- Add the chosen dataset to existing graph

5.3.3 Steps

Choosing the Data Source

1. Click the **New Project** button on the Standard toolbar, to begin with a new project.
2. Click the **Import Wizard** button on the Standard toolbar. The Import Wizard opens. (Note that if this is the first time that you have started the Import Wizard, you will experience a slight delay as Origin compiles the necessary files.)
3. Verify that the **ASCII** radio button is selected in the **Data Type** group.
4. Click the browse button to the right of the File text box. Navigate to the Origin folder; browse to the **Samples** folder and then the **Import and Export** folder.
5. Double-click to select S15-125-03.dat from the list of files. Repeat for files, S21-235-07.dat and S32-014-04.dat.
6. Click **OK**.
7. Leave the **Import Filters for current Data Type** as **Data Folder: VarsFromFileNameAndHeader**. (This filter has the settings to use when importing the file.)
8. Set the **Import Mode** as **Start New Sheets**.
9. Click the **Finish** button. The three data files import into the workbook, each as a new sheet. You will have a book with three sheets: **Trial Run 1**, **Trial Run 2**, and **Trial Run 3**.

Plotting the Data

1. Select **Trial Run 1** sheet.
2. Highlight the D(Y) column.
3. Click the **Line** button on the **2D Graphs** toolbar. A new graph is created.

Adding data to the graph

1. Double-click on the layer 1 icon in the upper-left hand corner of the graph. The Plot Setup dialog opens.
2. Select **Layer 1** in the Plot List.
3. Click the blue arrows in the upper right corner of the dialog to **Show Plot Designations**.
4. Again click the blue arrows in the upper right corner of the dialog to **Show Available Data**.
5. Select **Trial Run 2** from the Available Data list.
6. Check **Time** as X and **Position** as Y.
7. Click **Add**.
8. Check the **Rescale** checkbox.
9. Click **OK**.

Updating the Legend and Formatting the Plot

1. Select **Graph:New Legend**.
2. Double-click on the line symbol for the second data plot in the legend. The Plot Details dialog opens.

3. Change the **Color** from Black to Red.
4. Click **OK**.

5.4 Add multiple fitted curves in a Histogram

5.4.1 Summary

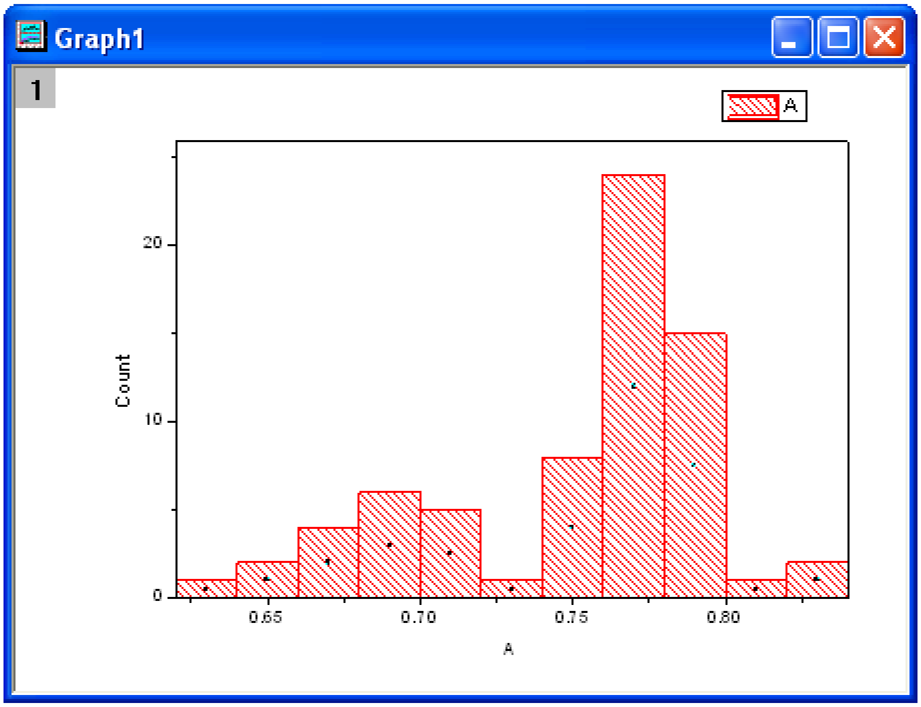
After you plot a Histogram, Origin allows you to overlay a distribution curve on the binned data by selecting **Normal**, **Lognormal**, **Poisson**, **Exponential**, **Laplace**, or **Lorentz** from the **Type** drop-down list in the Data tab of the Plot Details dialog. If you want to add multiple distribution curves to the Histogram, the procedure involves a few more steps.

5.4.2 What you will learn

- Plotting the Histogram
- Using Frequency Count to do statistics
- Using Peak Analyzer to find peaks and do fitting
- Adding new layers

5.4.3 Steps

Copy and paste the sample data into Origin and set the object column as Y. Plot this data as a Histogram by clicking **Plot: Statistics: Histogram** from the menu.

1. 

Frequency Count

1. Highlight the sample data, then open the **Frequency Count** dialog by selecting **Statistics: Descriptive Statistcs: Frequency Count**.

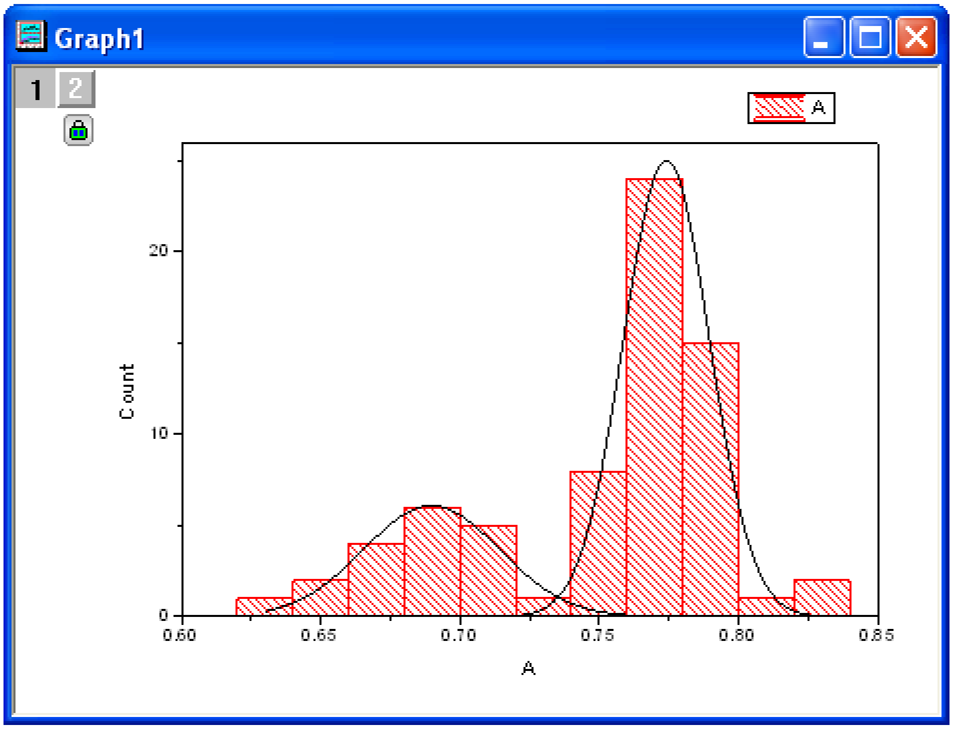
2. Click **OK** to finish. A new result sheet will be generated.

Fit peaks

1. Select Col(Counts) and then open the **Peak Analyzer** dialog from **Analysis: Peak and Baseline: Peak Analyzer**.
2. In the start page, select **Fit Peaks** as **Goal**, then click **Next**.
3. In the **Baseline Mode** page, set the baseline as **Y=0** when the **Custom** radio box is checked.
4. Click the **Next** button twice to go to the **Find Peaks** page. Click the **Find** button under the **Enable Auto Find** check box to find two peaks.
5. Click the **Next** button again to open the **Fit Peaks** page. Click **Fit Control** at the bottom of this page to open the **Peak Fit Parameters** dialog.
6. In this dialog, the default fitting function is **Gaussian**, which is the right function for normalizing the data. Close the **Peak Fit Parameters** dialog and go back to the **Peak Analyzer** dialog. Click **Finish** to complete the fitting.

Add the fitted curves

1. Active the Histogram graph and add a layer by selecting **Graph: New Layer(Axes): Right-Y** from the main menu.
2. Right-click the **Layer2** icon and select **Plot Setup** from the short menu to open the **Plot Setup** dialog.
3. Select the sheet **FitPeakCurve1** from the top panel, then set col(C1) as X and col(C2) as Y, and add them into the Layer **RightY** in the bottom panel.
4. Do the same things for col(C3) and col(C4). After that, both fitted peaks have been added into the Layer **RightY**.
5. Click OK. Two fitted curves had been added to the Histogram.
6. Double-click the graph to open the **Plot Details** dialog. Select **RightY** from the left panel, then open the **Link Axes Scales** tab in the right panel and select **Straight(1 to 1)** for both **X Axis Link** and **Y axis Link**. Click OK to close the dialog.
7. The fitted curves are added into the Histogram with the proper scale. The following is the result graph, with the right Y-axis removed.



Sample Data

0.631
0.642
0.652
0.662
0.669
0.676
0.677
0.69
0.691
0.696
0.697
0.699
0.699
0.7
0.7
0.708
0.712
0.718
0.731
0.744
0.749
0.751
0.752

0.753
0.758
0.758
0.759
0.761
0.761
0.763
0.763
0.763
0.765
0.767
0.768
0.768
0.769
0.769
0.77
0.771
0.771
0.772
0.774
0.775
0.775
0.776
0.776
0.776
0.777
0.778
0.779
0.78
0.78
0.781
0.784
0.784
0.785
0.785
0.789
0.789
0.791
0.794
0.795
0.796
0.798
0.798

0.803
0.82
0.831

5.5 Copy and Apply a Plot Format to Another Plot

5.5.1 Summary

It is possible to copy and paste formatting from one plot to another, so there is no need to spend time recreating identical customizations such as size and color of symbols and lines.

Minimum Origin Version Required: Origin 8.0 SR6

5.5.2 What you will learn

This tutorial will show you how to:

- Copy a plot format (color, size, etc. of the symbol or the line), and apply it to other plot.

5.5.3 Steps

1. Click the **New Project** button on the Standard toolbar to begin with a new project.
2. Select **File: Import: Single ASCII** menu, and import **exponential decay.dat** in the **\Samples\Curve Fitting** subfolder in your Origin program directory.
3. Highlight column B, C and D, and select **Plot: Line+Symbol: Line+Symbol" menu to plot these three datasets.**
4. Double-click on the plot to show the **Plot Details** dialog box.
5. Choose **Group** tab in the dialog, and select **Independent** for the **Edit Mode** -- this makes it easier to customize individual plots.
6. Make sure that the top data plot (Time(X) Decay 1(Y)) is selected in the left panel of the **Plot Details** dialog. If not, select this plot branch in the left panel.
7. Select the **Symbol** tab, and change the Size to "5". (You can also change the shape or the color to others of your choice.)
8. Select the **Line** tab, and change the width to "0.2". (You can also change the style or the color to others of your choice.) Click OK. You will see that the **Decay 1** plot has been customized.
9. Click on **Decay 1** plots to select. Right-click on it and select **Copy format: ALL**. This will copy the plot format of Decay 1 to the clipboard.
10. On the graph, click on the **Decay 2** data plot to select it, right-click and select **Paste Format**. You will see the plot format of Decay 1 copied to Decay 2.

5.6 Adding and Arranging Layers

5.6.1 Summary

A typical graph page generally includes three elements: a set of X, Y (and Z) coordinate axes, one or more data plots, and associated text and graphic labels. Origin combines these three elements into a movable, resizeable unit called a layer. While a page can contain as many as 121 layers, only one layer can be active at any one time.

5.6.2 What you will learn

- How to tell how many layers a graph has?
- How to tell what data is in what layer?
- How to swap layers

5.6.3 Steps

Importing data

1. Click the **Import Single ASCII** button, , on the Standard toolbar. The ASCII dialog opens.
2. In the Origin folder, browse to the Samples folder and then the Graphing folder. Select Wind.dat from the list of files.
3. Click **Open**. The data file imports into the worksheet.

Plotting the data

1. Highlight the Speed and Power columns.
2. Click the **Line** button, , on the **2D Graphs** toolbar. A line plot is created. It appears that this data would be better plotted on a double-Y graph, a graph with two controlling Y axes.
3. Click the X to **Close** this window. You will be asked if you want to hide or delete the window. Click the **Hide** button. (If you Delete, you will not be able to Undo, and you will need to re-create the graph. Hiding closes the window from view, but you can later make it visible using the Project Explorer window.)
4. The Speed and Power columns should still be highlighted. Click the **Double-Y Axis** button, , on the 2D Graphs toolbar. This new graph contains 2 layers.

How to tell what data is plotted in a layer?

One way is with the legend:

1. Double-click on Graph1 from the Project Explorer window. The graph opens and becomes the active child window.
2. Select **Format:Page** and go to the **Legends/Titles** tab.
3. Set the **Auto Legend Translation Mode** to *Data Range*.
4. Click **OK**.

A second way is with the status bar:

1. Double-click on Graph2 from the Project Explorer window. Graph2 is now in front and becomes the active child window.
2. Click the Layer 1 icon, 1 to make layer 1 active.
3. Look in the lower right of Origin's status bar and you'll see [WIND]WIND!Col(Speed)[1:12].
4. Repeat for Layer 2 and you'll see [WIND]WIND!Col(Power)[1:12].

A third way is from the menu:

1. Make Graph2 active.
2. Right-click on the Layer 1 icon. At the bottom of the context menu, you see the data plot list. The one checked is the active data plot.
3. Right-click on Layer 2 to view the list of data that is plotted in it.
4. Note that you can also select the **Data** menu to view the data plot list.

A fourth way is with the **Plot Setup** dialog.

1. With Graph2 still active, double-click on the layer 1 icon.
2. In the opened Plot Setup dialog, expand the Plot List to see also the data in Layer 2. You can see the advantage here is that you can see data in all layers at once.
3. Uncheck the Show check box for the **Speed** data plot.
4. Click **OK**. This data plot is only hidden which is why the legend still indicates its presence in the layer.

The last way is from the **Plot Details** dialog.

1. Double-click on any of the line+symbol plots in Graph2.
2. Expand the tree so you see contents of both **Layer1** and **Layer2**.
3. Uncheck the **Speed** and **Power** data plots.
4. Click **OK**.

5.7 Create an 8 layer multi-panel plot template

5.7.1 Summary

All child windows in Origin, with the exception of the Notes window, are created from template files. These template files describe how to construct the window. For a graph window, the template file determines all page and layer characteristics, such as page size, number of layers, inclusion of text labels, data plot style information, etc.

The template library lists all built-in as well as user-created templates.

Minimum Origin Version Required: Origin 8.0 SR6

5.7.2 What you will learn

- How to create an 8 layer multi-panel plot
- How to save the formatting as a template
- How to reuse the template with similar data

5.7.3 Steps

Choosing the Data Source

1. Click the **New Project** button on the Standard toolbar, to begin with a new project.
2. Click the **Import Wizard** button on the Standard toolbar. The Import Wizard opens. (Note that if this is the first time that you have started the Import Wizard, you will experience a slight delay as Origin compiles the necessary files.)
3. Verify that the **ASCII** radio button is selected in the **Data Type** group.
4. Click the browse button to the right of the File text box. Navigate to the Origin folder; browse to the Samples folder and then the Curve Fitting folder. Select Step01.dat from the list of files.
5. Click the **Add File(s)** button.
6. Click **OK**.
7. Leave the **Import Filters for current Data Type** as **Data Folder: step**. (This filter has the settings to use when importing the file.)
8. Click the **Finish** button. The data file imports into the worksheet.

Plotting the Data

1. Highlight the entire worksheet of data. (Note that you can select an entire worksheet by placing your cursor in the blank area in the upper left corner of the worksheet. When the cursor becomes a downward pointing arrow, click once to select the entire worksheet.)
2. Select **Plot:Multi-Curve:9 Panel**. A new 9 layer graph is created.
3. Select layer 9 by clicking inside it.
4. Press the **Delete** key on the keyboard. This will delete layer 9, leaving you with 8 layers.
5. Select **Graph:Layer Management**. The **Layer Management** dialog opens.
6. Select the **Arrange** tab.
7. Set **Column** to 2 and **Row** to 4.
8. Click the **Apply** button. The preview in the dialog redraws to show you a 2x4 arrangement.
9. Click **OK**.

Editing the Graph

The goal is to save this 8 panel graph as a template; i.e. a new plot type, so that it can be used again with new similar data. Since the template will also save plot style information, let's customize the graph a bit further.

1. Double-click on the X axis in layer 1. The **X-Axis** dialog opens.
2. Select the **Grid Lines** tab.
3. Check the **Major Grids** and **Minor Grids** checkboxes.
4. Set the **Line Color** for both Major and Minor grids to **LT Gray**.
5. Check the **Apply To Grid Lines** checkbox for **This Layer**.
6. Click OK.
7. Select layer 1 by clicking inside it.
8. Right-click inside the layer and select **Copy Format:All Style Formats**. This will copy the style formats of layer 1.

9. To apply formatting to all layers, right-click outside of any layer (make sure that no layer is selected...one easy way to do that is to right-click in the gray area of the window, outside the white printable part of page), and select **Paste Format**.

Saving as a new graph template

1. Select **File:Save Template As**. A dialog opens allowing you to choose the category that the template will be saved in as well as the name given to the new template.
2. Change the **Category** to **UserDefined**.
3. Change the **Template Name** from **PAN9** to **PAN8**. (Note that the **Template Name** that appears when the dialog opens is the name of the original template that was used to create the graph.)
4. Expand the **Option** node and note that the template will be saved to your **User Files Folder**.
5. In the **Description** edit box, enter the following: My new 8 panel graph
6. Click OK.

Plotting into your new template

1. Click the **New Folder** button on the Standard toolbar.
2. Click the **Import Wizard** button on the Standard toolbar. The Import Wizard opens.
3. Click the browse button to the right of the File text box. Navigate to the Origin folder; browse to the Samples folder and then the Curve Fitting folder. Select Step02.dat from the list of files.
4. Click the **Add File(s)** button.
5. Click **OK**.
6. Leave the **Import Filters for current Data Type** as **Data Folder: step**. (This filter has the settings to use when importing the file.)
7. Click the **Finish** button. The data file imports into the worksheet.
8. Highlight the entire worksheet of data. (Note that you can select an entire worksheet by placing your cursor in the blank area in the upper left corner of the worksheet. When the cursor becomes a downward pointing arrow, click once to select the entire worksheet.)
9. Select **Plot:Template Library** or click the **Template Library** button on the **2D Graphs** toolbar.
10. Scroll down to the **UserDefined** category under Graph Template.
11. Select **PAN8**. (Note that the Preview window is not a preview of the new data that you are plotting. It is an image of the graph when you saved your template.)
12. Click **Plot**.

5.8 Mark out a segment of plot with different plot style

5.8.1 Summary

In Origin, you can mark out a segment of plot with different plot style, such as a segment of dashed in a solid line plot.

Minimum Origin Version Required: Origin 8.0 SR6

5.8.2 What you will learn

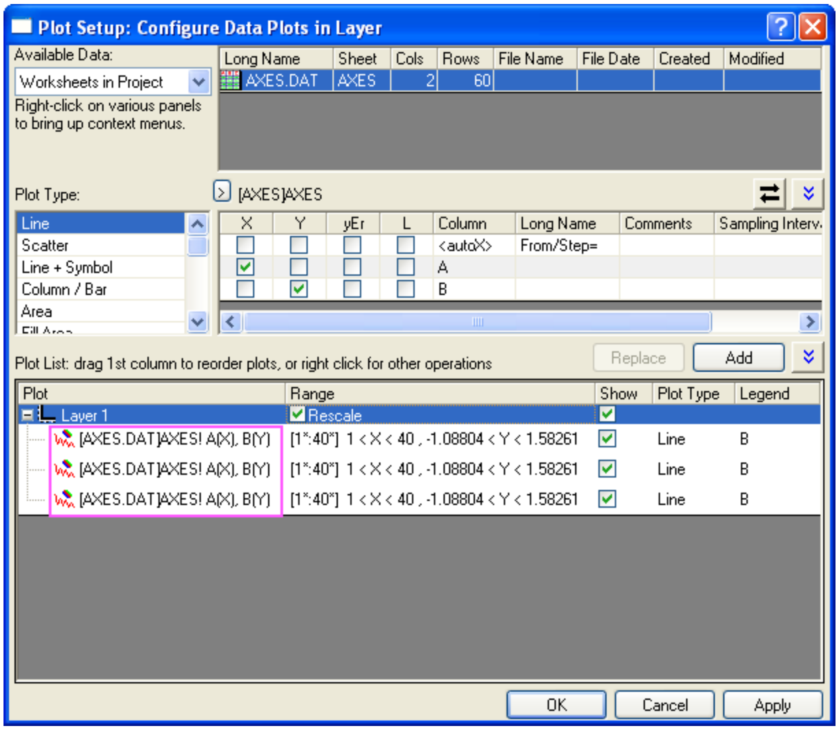
- Use the **Plot Setup** dialog to create a graph
- Mark out the special segment of a plot

5.8.3 Steps

1. Start with an empty worksheet. Select **File: Import: Single ASCII...** from the Origin menu to open the **Import Single ASCII** dialog. Browse to the *\Samples\Graphing* subfolder of the Origin program folder, and import the file *AXES.DAT* .

2. Click the button on the **Standard** toolbar to create a new graph window and then select **Graph: Plot Setup** from the main menu to bring up the **Plot Setup** dialog.

3. Show all of the three panels of the **Plot Setup** dialog. Select the **AXES** worksheet in the top panel. Then go to the middle panel to select **A** as X and **B** as Y. After that, click **Add** to add this data plot to the bottom panel. Repeat this step three times. Three data plots should be listed in the bottom panel.



Right-click on various panels to bring up context menus.

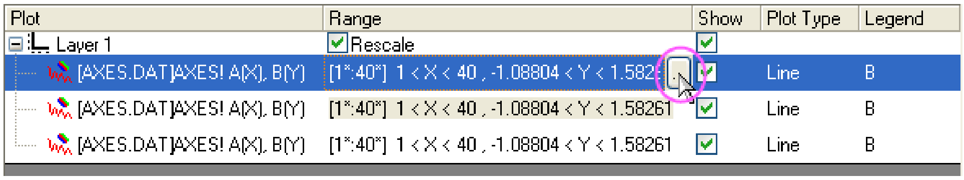
Long Name	Sheet	Cols	Rows	File Name	File Date	Created	Modified
AXES.DAT	AXES	2	60				

X	Y	yEr	L	Column	Long Name	Comments	Sampling Interv.
□	□	□	□	<autoX>	From/Step=		
☑	□	□	□	A			
□	☑	□	□	B			

Plot List: drag 1st column to reorder plots, or right click for other operations

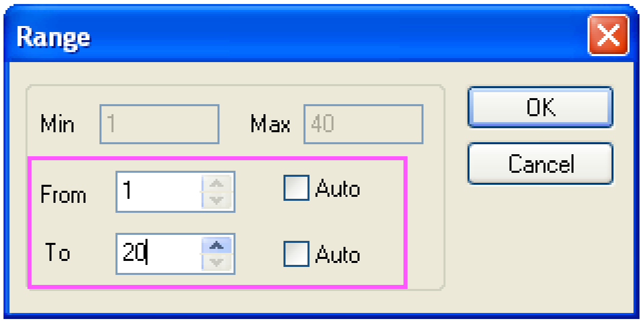
Plot	Range	Show	Plot Type	Legend
Layer 1	☑ Rescale	☑		
[AXES.DAT]AXES! A(X), B(Y)	[1*:40*] 1 < X < 40 , -1.08804 < Y < 1.58261	☑	Line	B
[AXES.DAT]AXES! A(X), B(Y)	[1*:40*] 1 < X < 40 , -1.08804 < Y < 1.58261	☑	Line	B
[AXES.DAT]AXES! A(X), B(Y)	[1*:40*] 1 < X < 40 , -1.08804 < Y < 1.58261	☑	Line	B

4. In the lower panel, click in the **Range** column that corresponds to the first data plot. The ... button should be activated. Then click this button to open the **Range** dialog box.



Plot	Range	Show	Plot Type	Legend
Layer 1	☑ Rescale	☑		
[AXES.DAT]AXES! A(X), B(Y)	[1*:40*] 1 < X < 40 , -1.08804 < Y < 1.582 ...	☑	Line	B
[AXES.DAT]AXES! A(X), B(Y)	[1*:40*] 1 < X < 40 , -1.08804 < Y < 1.58261	☑	Line	B
[AXES.DAT]AXES! A(X), B(Y)	[1*:40*] 1 < X < 40 , -1.08804 < Y < 1.58261	☑	Line	B

5. Clear the **Auto** check boxes (if they are selected) and then set **From** to **1** and **To** to **20**. Click **OK** to close the dialog.

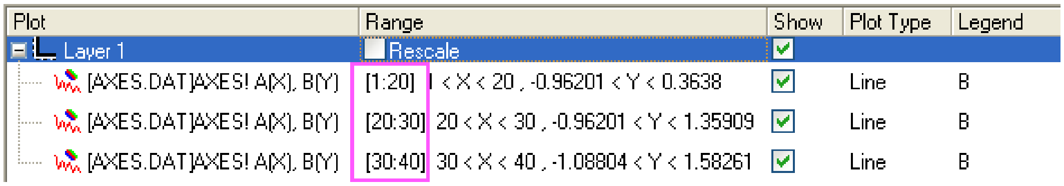


Min 1 Max 40

From 1 ☐ Auto

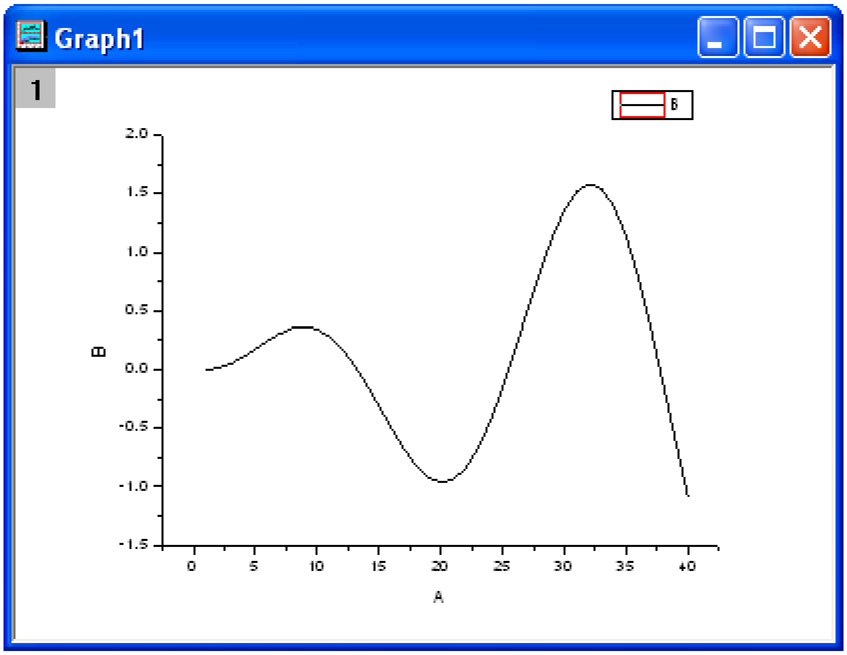
To 20 ☐ Auto

6. Similarly, set the ranges for other two data plots to "20 to 30" and "30 to 40" respectively.

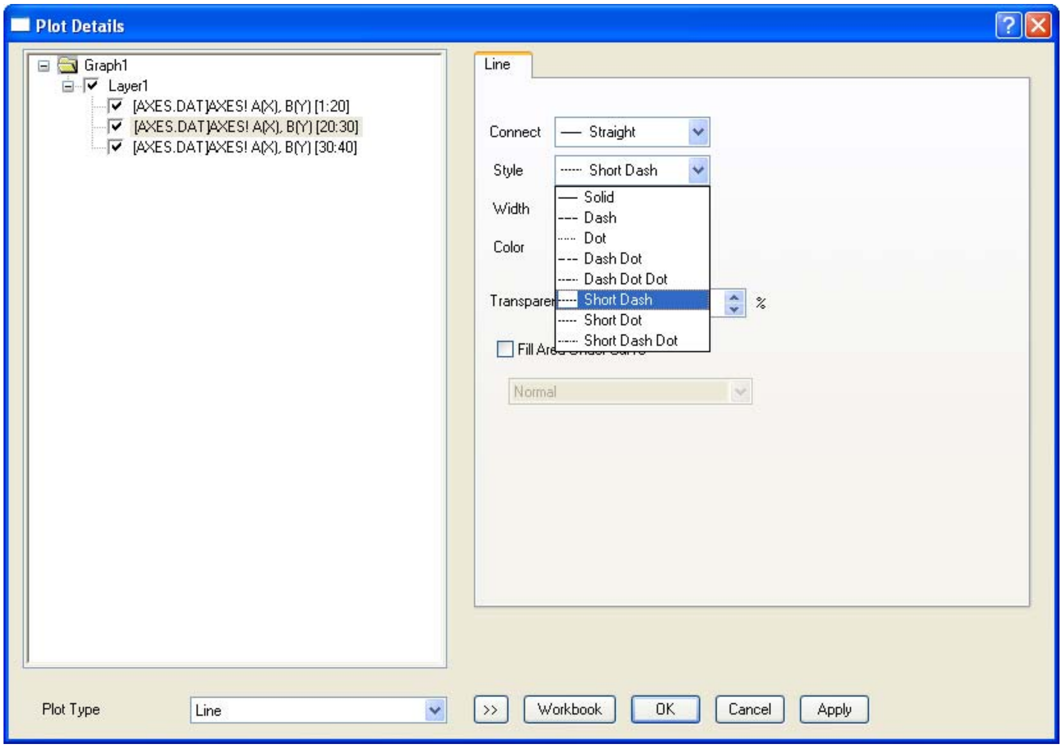


Plot	Range	Show	Plot Type	Legend
⊟ Layer 1	☐ Rescale	☑		
[AXES.DAT]AXES! A(X), B(Y)	[1:20] 1 < X < 20 , -0.96201 < Y < 0.3638	☑	Line	B
[AXES.DAT]AXES! A(X), B(Y)	[20:30] 20 < X < 30 , -0.96201 < Y < 1.35909	☑	Line	B
[AXES.DAT]AXES! A(X), B(Y)	[30:40] 30 < X < 40 , -1.08804 < Y < 1.58261	☑	Line	B

7. Click the **OK** button to close the **Plot Setup** dialog. You should get a graph like this:



8. Double-click on the curve in the graph window to open the **Plot Details** dialog. Select the second data plot from the left panel. In the right panel, change the **Style** to **Short Dash**, and then click the **Ok** button.



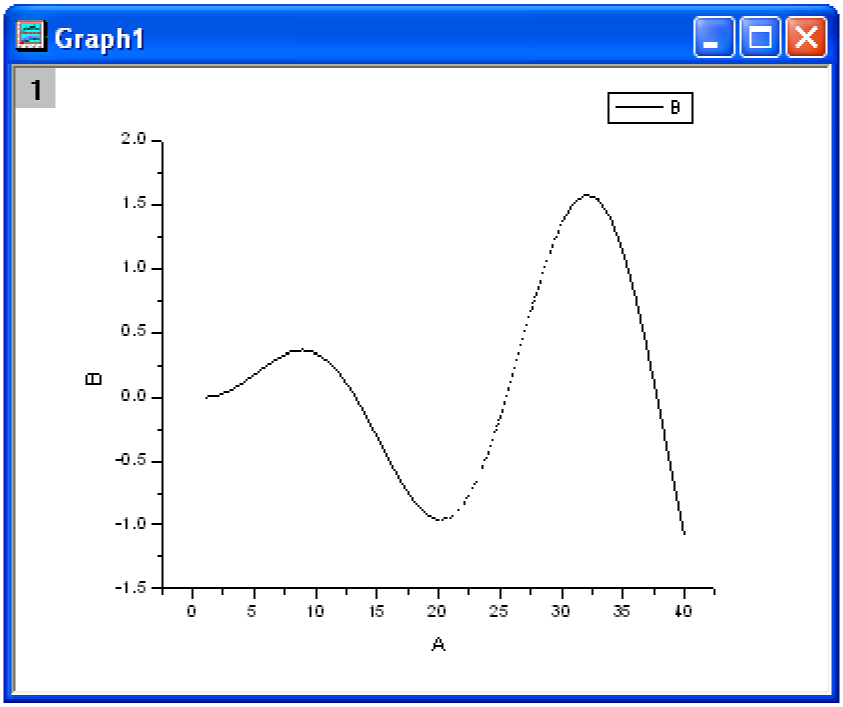
Plot Details

Connect: Straight

Style: Short Dash

Plot Type: Line

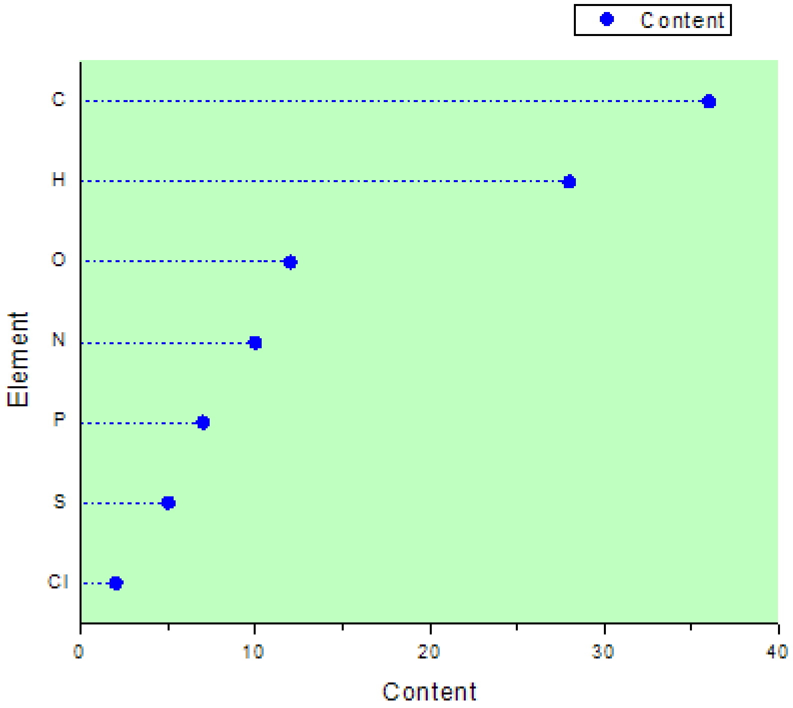
9. Finally, we get the plot with a range marked out.



5.9 Simple Dot Chart

5.9.1 Summary

Dot chart is a statistical chart which consist of data points plotted on a simple scale. It is often used as a substitute for the pie chart, as it allows for quantities to be compared easily. This tutorial will teach you how to create a simple dot chart plot.



Minimum Origin Version Required: Origin 8.0 SR6

5.9.2 What you will learn

This tutorial will show you how to:

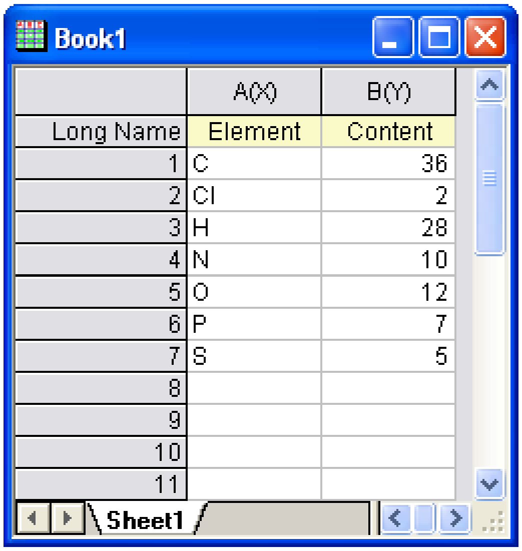
- Create a scatter graph
- Change the X-Y Axis
- Use Plot Setup dialog to customize your graph

5.9.3 Steps

Let us start with the following data which represents various elements in a compound:

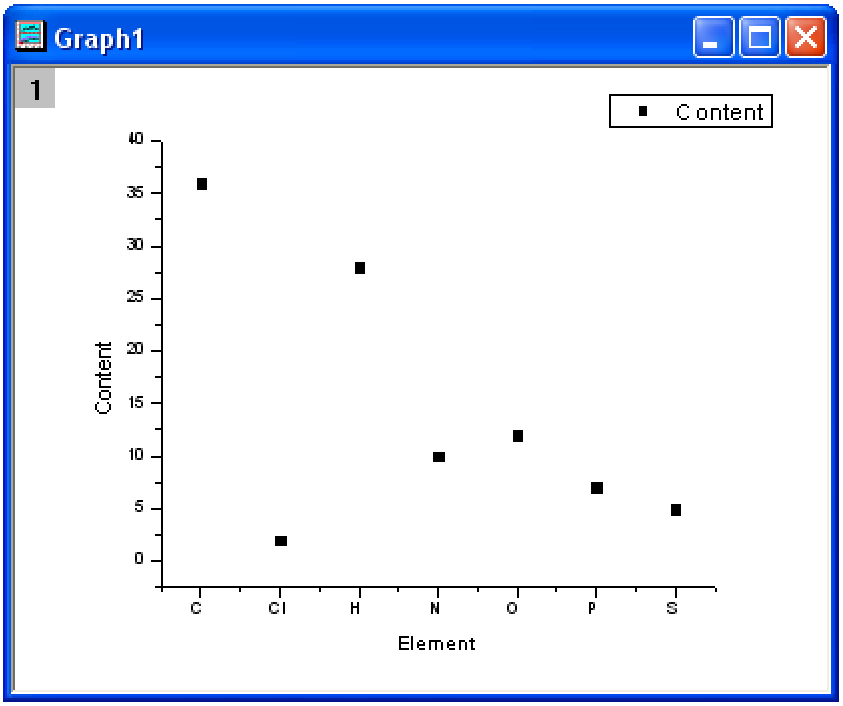
Element	Content
C	36
Cl	2
H	28
N	10
O	12
P	7
S	5

1. Create a new workbook, and input the data.

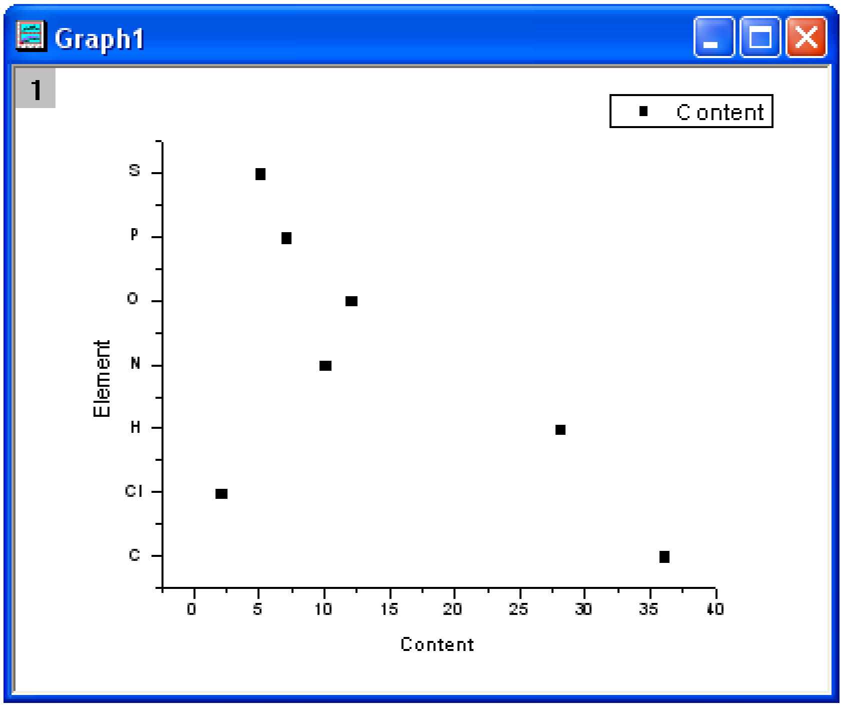


	A(X)	B(Y)
Long Name	Element	Content
1	C	36
2	Cl	2
3	H	28
4	N	10
5	O	12
6	P	7
7	S	5
8		
9		
10		
11		

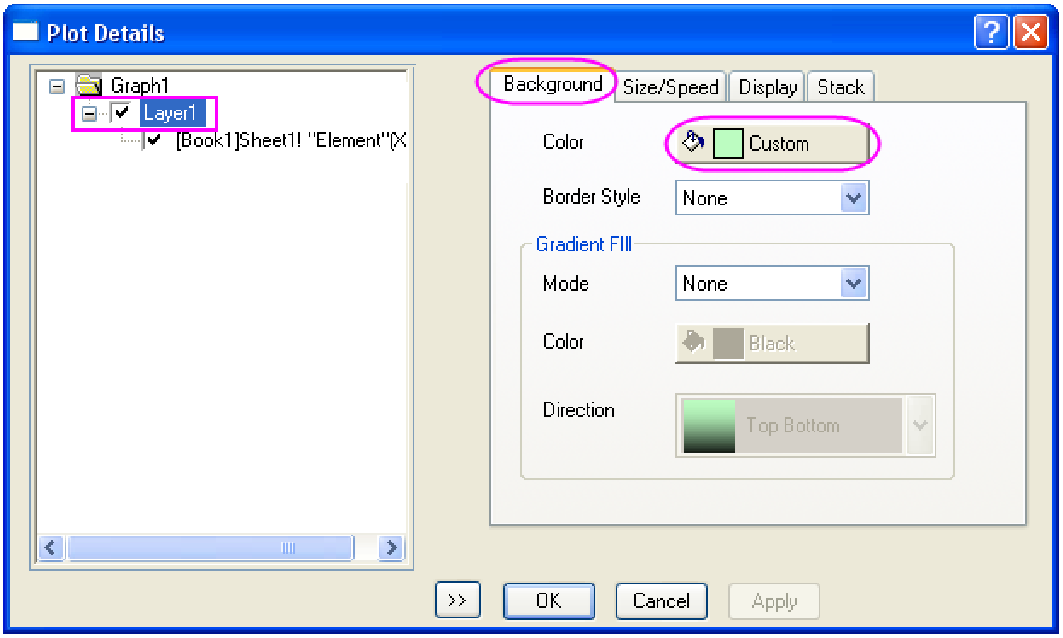
2. Highlight col(A) and col(B), and then select the **Plot: Symbol: Scatter** menu item from the Origin menu to create a scatter plot.



3. Select **Graph: Exchange X-Y Axis** from the menu.



4. Double-click on the graph to bring up the **Plot Details** dialog, change the symbols and the symbol color as in the following image:



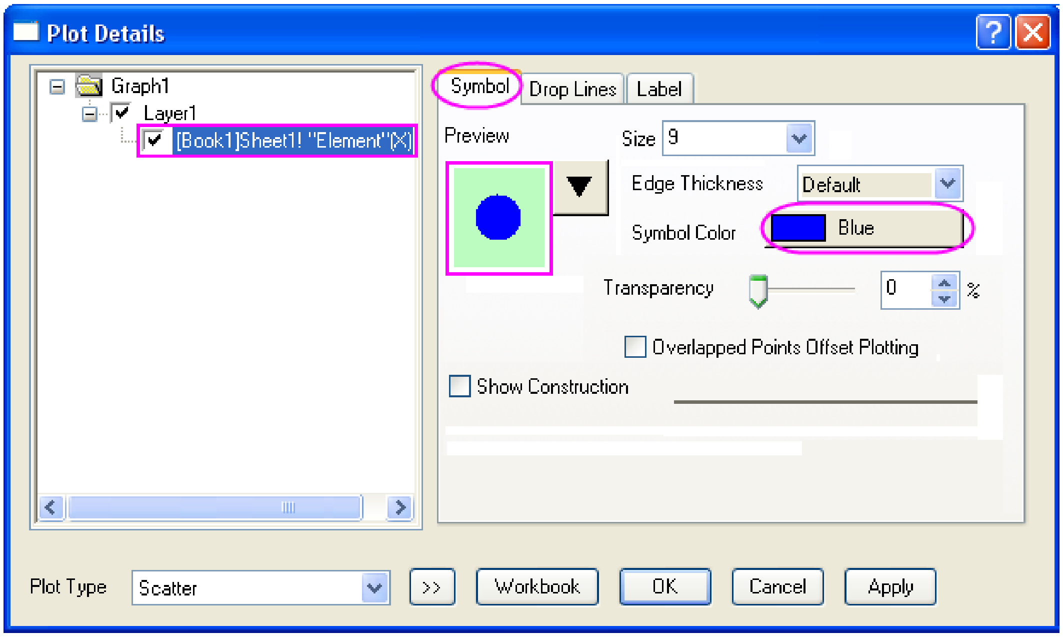
Color: 🟩 Custom

Border Style: None

Mode: None

Color: Black

Direction: Top Bottom



☑ [Book1]Sheet1! "Element"(X)

Size: 9

Edge Thickness: Default

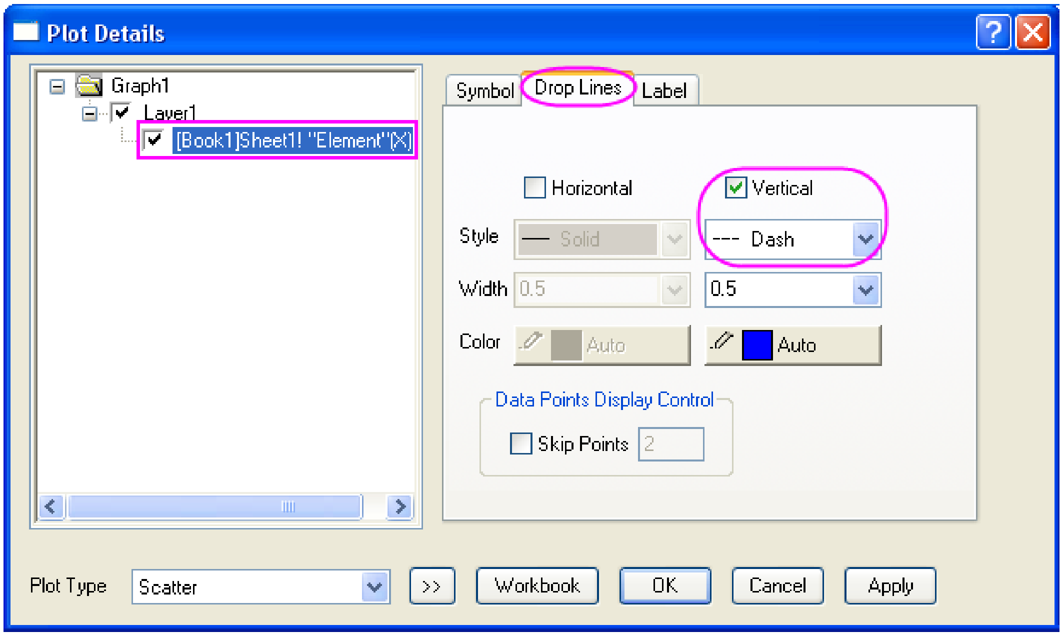
Symbol Color: Blue

Transparency: 0 %

☐ Overlapped Points Offset Plotting

☐ Show Construction

Plot Type: Scatter



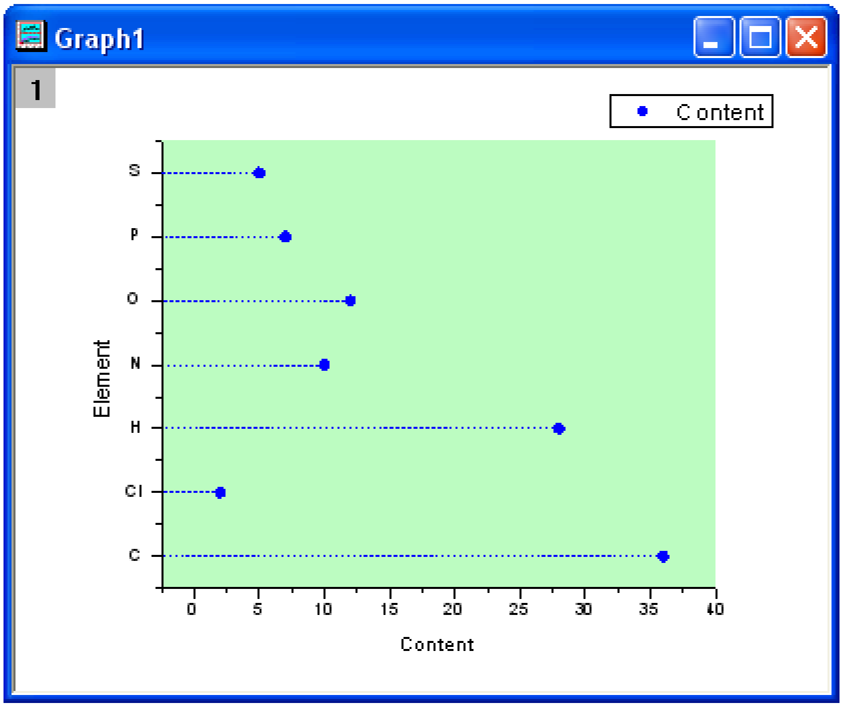
☑ [Book1]Sheet1! "Element"(X)

	☐ Horizontal	☑ Vertical
Style	Solid	Dash
Width	0.5	0.5
Color	Auto	Auto

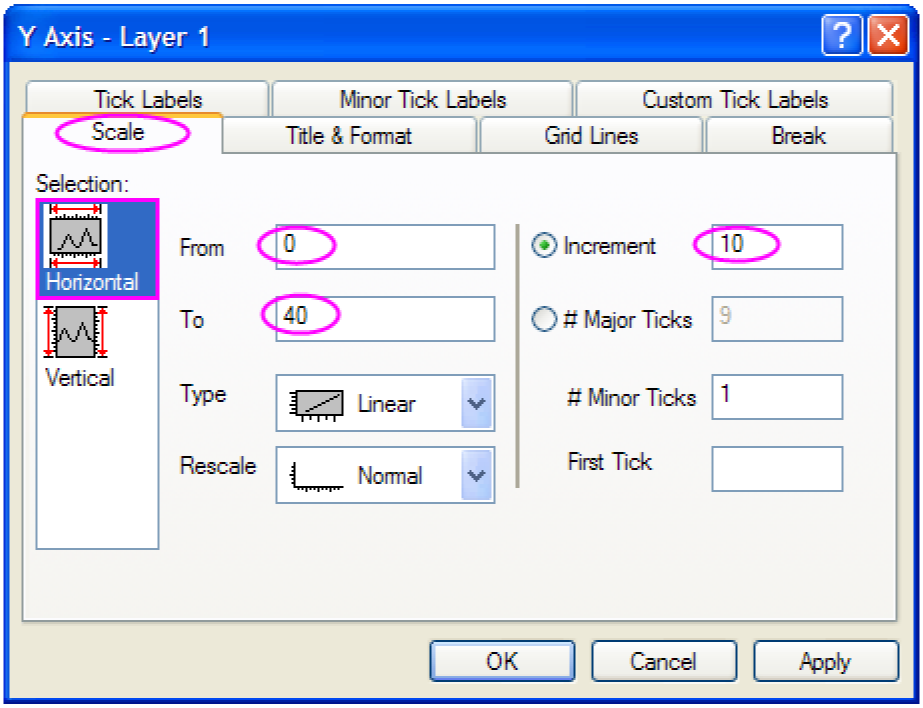
☐ Skip Points 2

Plot Type: Scatter

5. Click the **OK** button to close the dialog. Your graph should look like the image below:



6. Let us reset the X and Y Axes. Double-click the x axis to open the **Axis Properties** dialog. In the **Scale** tab, set **From** as **0** and **To** as **40**. Set the **Increment** as **10**.



From: 0

To: 40

Type: Linear

Rescale: Normal

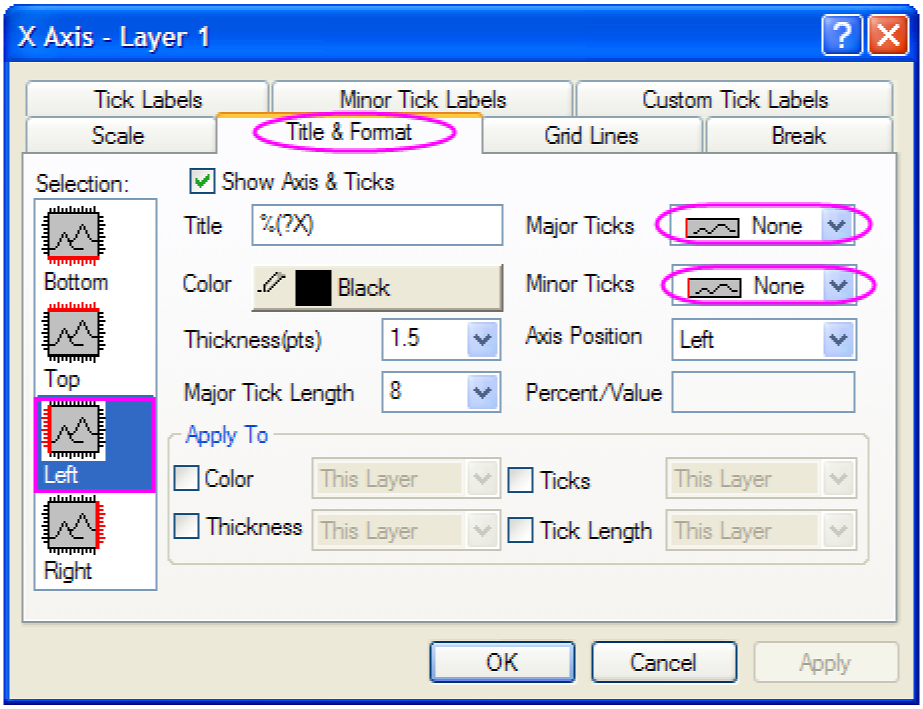
● Increment: 10

○ # Major Ticks: 9

Minor Ticks: 1

First Tick:

7. In the **Title and Format**, make sure **left** is selected in the **Selection** list, and then set **Major Ticks** and **Minor Ticks** as **None**. Finally, click **OK** button.



☑ Show Axis & Ticks

Title %(?X)

Color Black

Thickness(pts) 1.5

Major Tick Length 8

Major Ticks None

Minor Ticks None

Axis Position Left

Percent/Value

Apply To

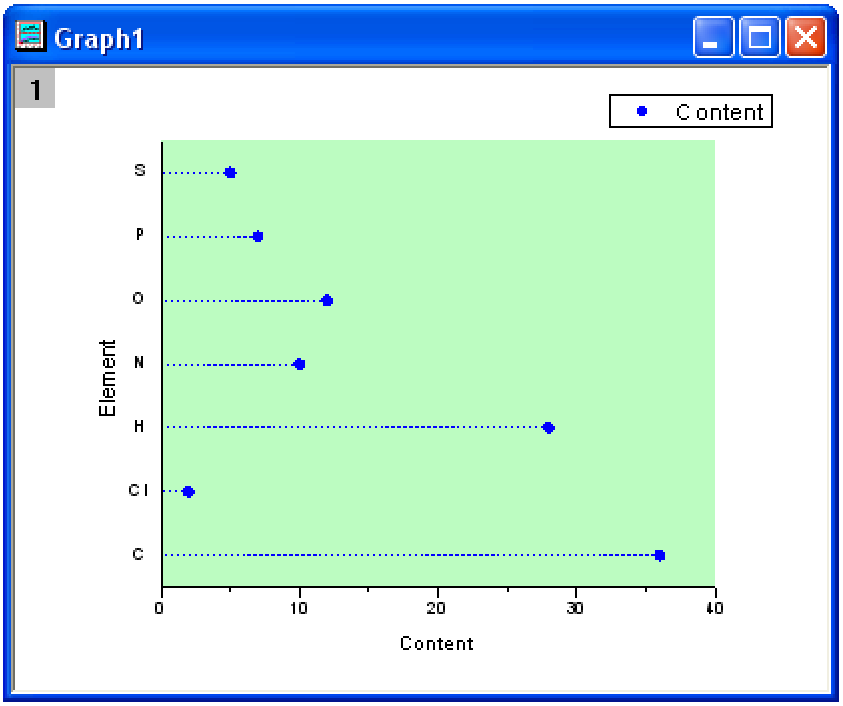
□ Color This Layer

□ Thickness This Layer

□ Ticks This Layer

□ Tick Length This Layer

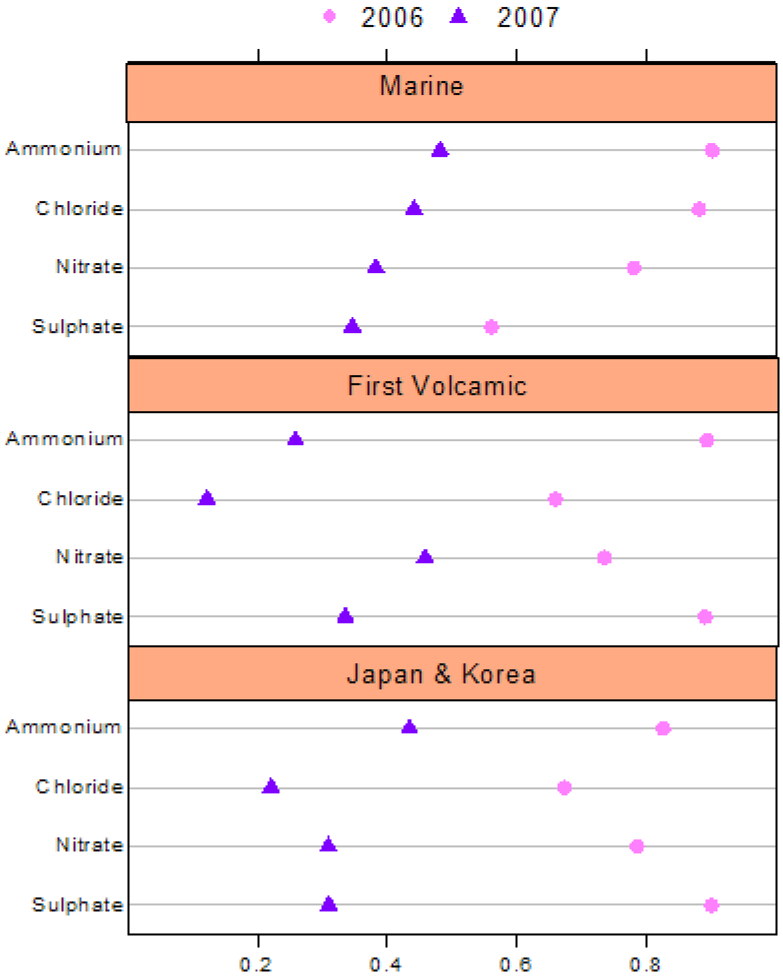
Now the Dot chart is plotted successfully and should look like below:



5.10 Multi-Data Dot Chart

5.10.1 Summary

A **Dot Chart** is a statistical chart which consists of data points plotted on a simple scale. It is often used as a substitute for the pie chart because it can make the comparing of quantities easy. This tutorial will teach you how to create the Multi-Data Dot Chart.



Minimum Origin Version Required: Origin 8.0 SR6

5.10.2 What you will learn

This tutorial will show you how to:

- Create a scatter graph
- Change the X-Y Axis
- Use the Plot Setup dialog to customize your graph
- Use Layer Management
- Customize the axis
- Add objects on the graph

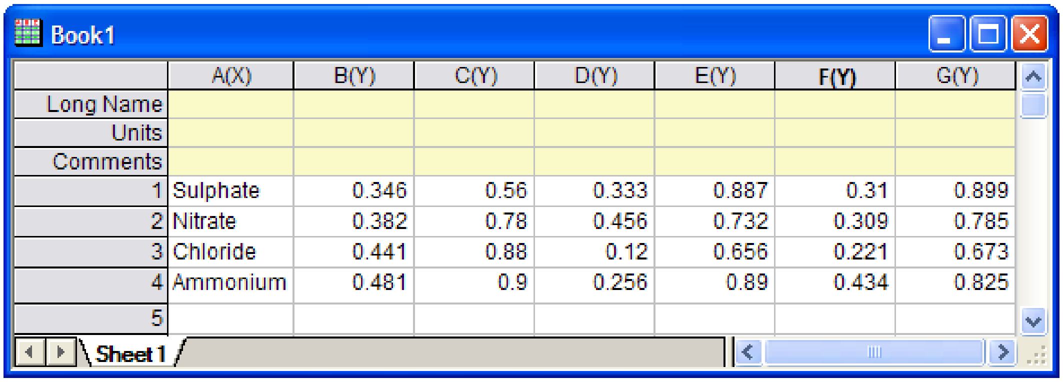
5.10.3 Steps

Let us learn how to create a multi-data dot chart. Here is some data about the element content of several areas in different time. We can use it to create a dot chart.

Sulphate	0.346	0.560	0.333	0.887	0.310	0.899
Nitrate	0.382	0.780	0.456	0.732	0.456	0.732
Chloride	0.441	0.880	0.120	0.656	0.221	0.673
Ammonium	0.481	0.900	0.256	0.890	0.434	0.825

Now, let us begin.

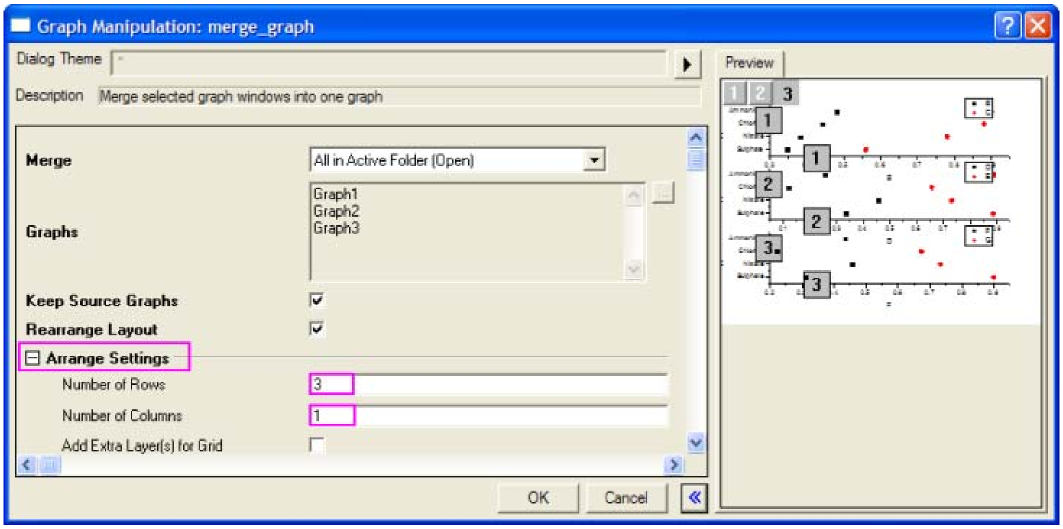
1. Create a new workbook and input the data.



Book1

	A(X)	B(Y)	C(Y)	D(Y)	E(Y)	F(Y)	G(Y)
Long Name							
Units							
Comments							
1	Sulphate	0.346	0.56	0.333	0.887	0.31	0.899
2	Nitrate	0.382	0.78	0.456	0.732	0.309	0.785
3	Chloride	0.441	0.88	0.12	0.656	0.221	0.673
4	Ammonium	0.481	0.9	0.256	0.89	0.434	0.825
5							

2. Highlight col(B) and col(C), select **Plot: Symbol: Scatter** in the main menu to draw a graph, then select **Graph: Exchange X-Y Axis**.
3. Repeat step 2 to create one graph with col(D) and col(E), and another graph with col(F) and col(G).
4. Merge these three graphs. Select **Graph: Merge Graph Windows: Open dialog**. Expand **Arrange Settings**, set **the Number of Rows** as **3** and **Number of Columns** as **1**. Click the **OK** button. Now you get a new graph which contains three layers.



Graph Manipulation: merge_graph

Dialog Theme: -

Description: Merge selected graph windows into one graph

Merge: All in Active Folder (Open)

Graphs: Graph1
Graph2
Graph3

Keep Source Graphs ☑

Rearrange Layout ☑

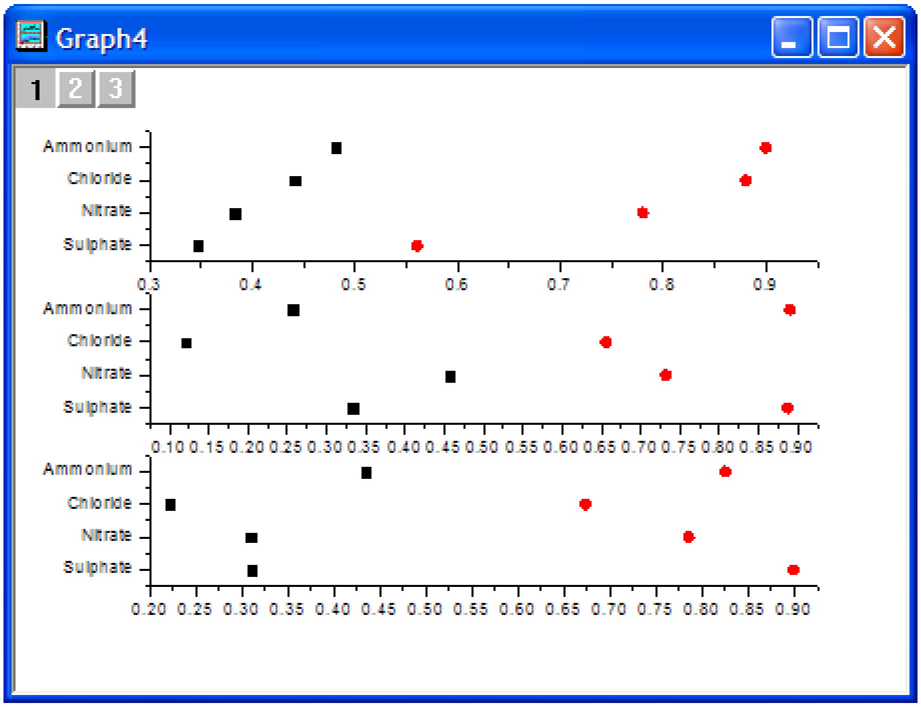
⊟ Arrange Settings

Number of Rows: 3

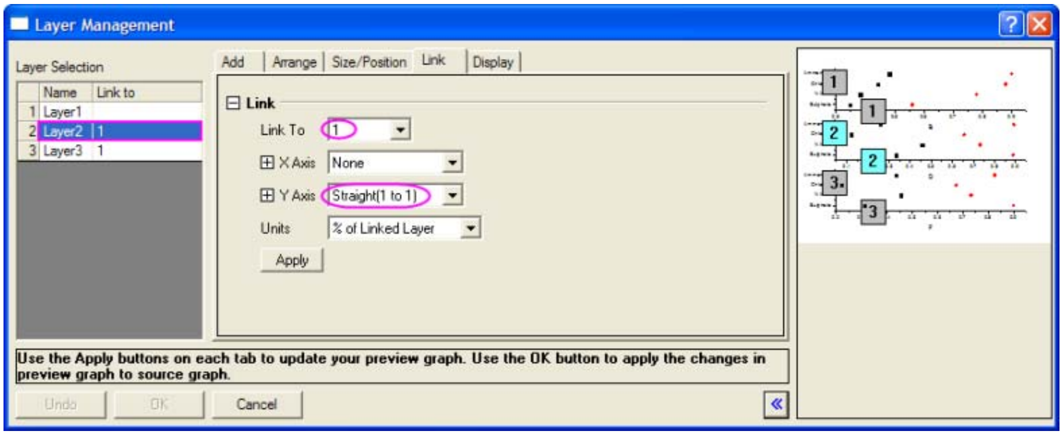
Number of Columns: 1

Add Extra Layer(s) for Grid □

5. You can delete the legend and the XY axis labels in the graphs to clean up the graph.



6. Select **Graph: Layer Management** from the main menu. Select Layer 2 on the left. Then on the right panel, make sure that the **Link** tab is active. In this tab, set **Link to** as **1** and **Y Axis** as **Straight (1 to 1)**. Click the **Apply** button.



	Name	Link to
1	Layer1	
2	Layer2	1
3	Layer3	1

⊟ Link

Link To: 1

⊞ X Axis: None

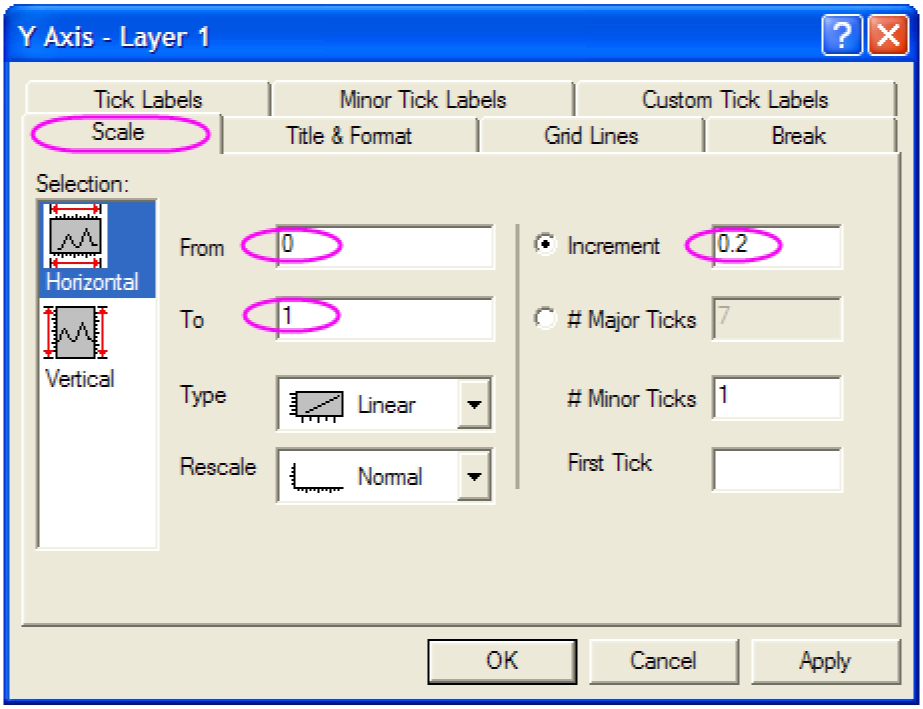
⊞ Y Axis: Straight(1 to 1)

Units: % of Linked Layer

Use the Apply buttons on each tab to update your preview graph. Use the OK button to apply the changes in preview graph to source graph.

7. Select Layer 3 on the left. Also in the **Link** tab, set **Link to** as **1** and **Y Axis** as **Straight (1 to 1)**. Then click the **OK** button.

8. Reset X and Y Axes. Make sure **Layer 1** is active, double-click the X axis of Layer 1. Set the scale of X **From 0 To 1**, and the **Increment** as **0.2**.



From 0

To 1

Type Linear

Rescale Normal

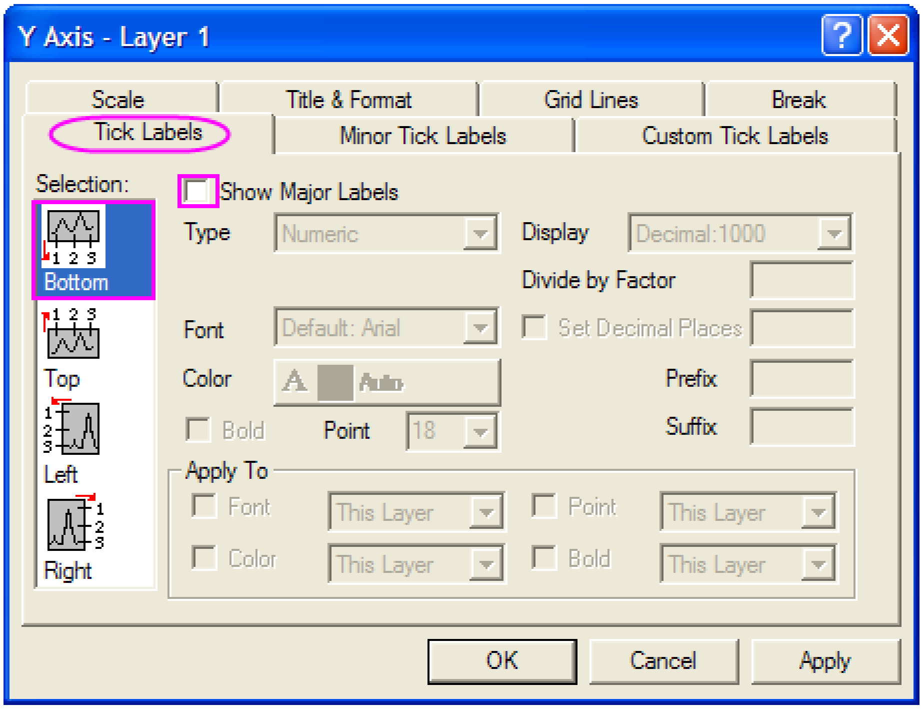
● Increment 0.2

○ # Major Ticks 7

Minor Ticks 1

First Tick

9. In the **Tick Labels** tab, uncheck the **Show Major Label** box.



□ Show Major Labels

Type Numeric

Display Decimal:1000

Divide by Factor

Font Default: Arial

□ Set Decimal Places

Color A Auto

Prefix

Suffix

□ Bold

Point 18

Apply To

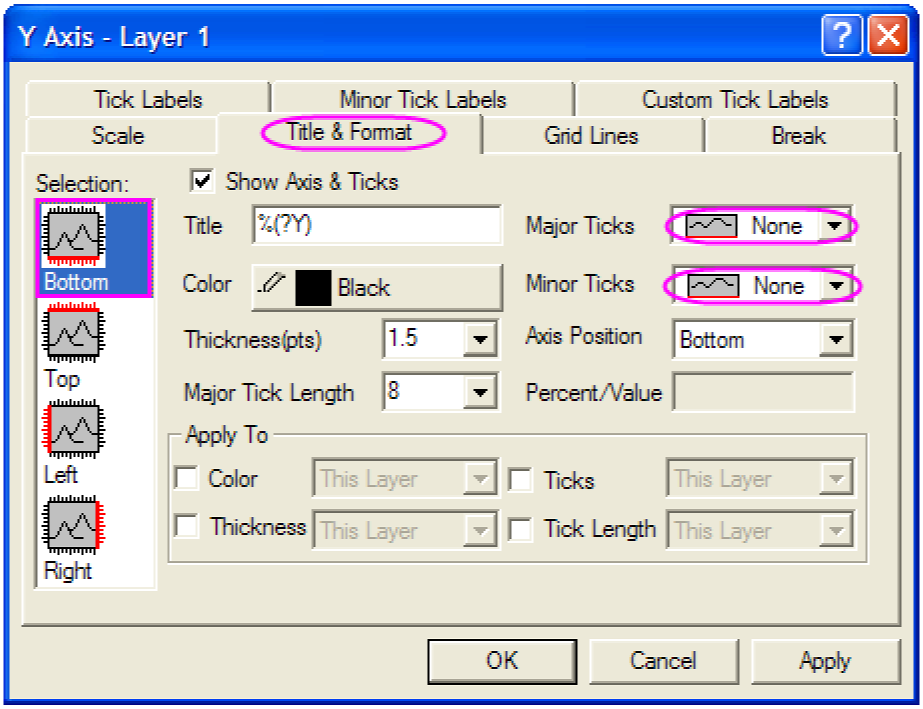
□ Font This Layer

□ Point This Layer

□ Color This Layer

□ Bold This Layer

10. In the **Title and Format** tab, set **Major Ticks** and **Minor Ticks** as **None**.



Y Axis - Layer 1

Selection: Bottom

☑ Show Axis & Ticks

Title: %(?Y)

Color: Black

Thickness(pts): 1.5

Major Tick Length: 8

Major Ticks: None

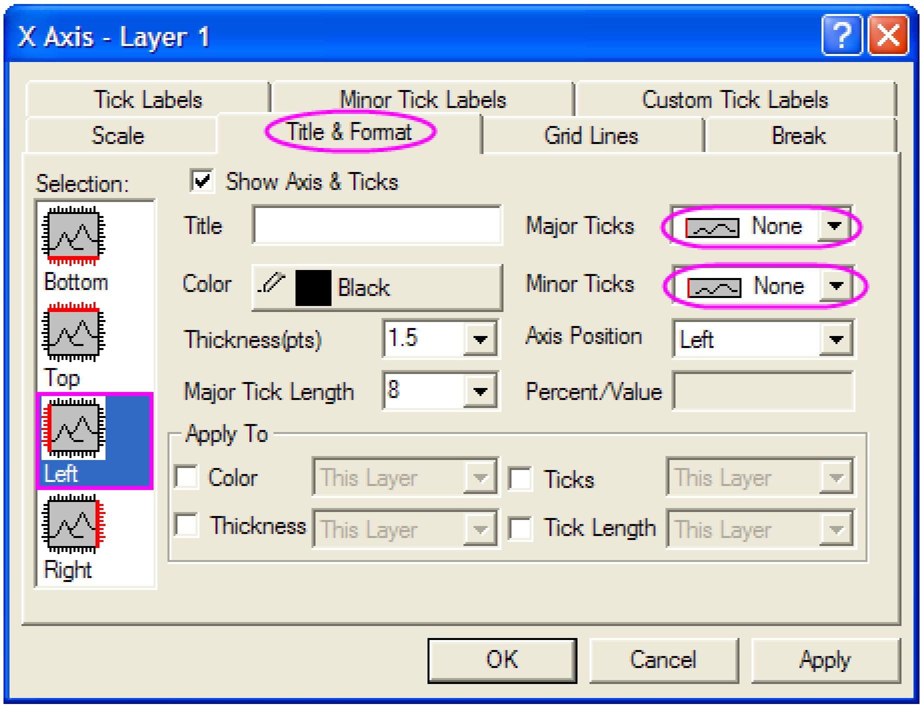
Minor Ticks: None

Axis Position: Bottom

Percent/Value:

Apply To: □ Color This Layer; □ Thickness This Layer; □ Ticks This Layer; □ Tick Length This Layer

11. Click **Left** in the **Selection** list, select the **Show Axis & Tick** check box and set **Major Ticks** and **Minor Ticks** as **None**.



X Axis - Layer 1

Selection: Left

☑ Show Axis & Ticks

Title:

Color: Black

Thickness(pts): 1.5

Major Tick Length: 8

Major Ticks: None

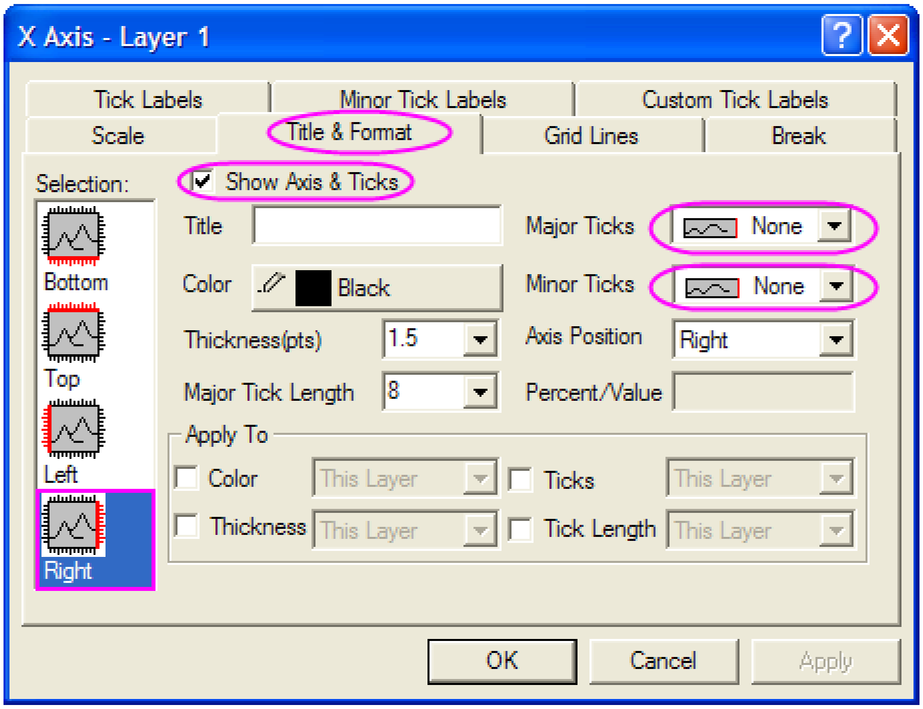
Minor Ticks: None

Axis Position: Left

Percent/Value:

Apply To: □ Color This Layer; □ Thickness This Layer; □ Ticks This Layer; □ Tick Length This Layer

12. Click **Right** in the **Selection** list, select the **Show Axis & Tick** check box and set **Major Ticks** and **Minor Ticks** as **None**.



X Axis - Layer 1

Selection: Right

☑ Show Axis & Ticks

Title:

Color: Black

Thickness(pts): 1.5

Major Tick Length: 8

Major Ticks: None

Minor Ticks: None

Axis Position: Right

Percent/Value:

Apply To

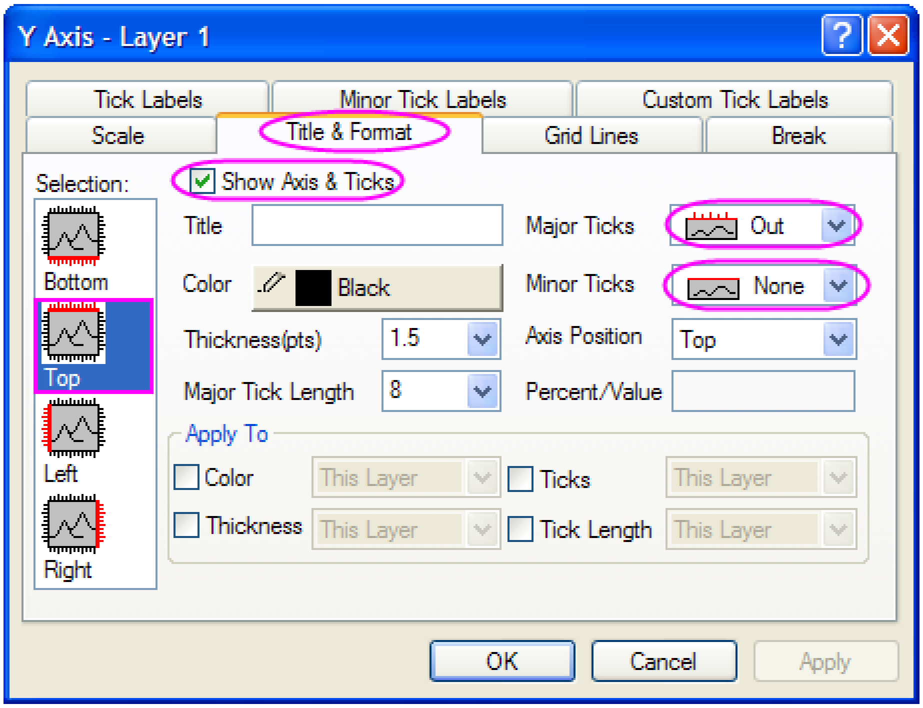
□ Color This Layer

□ Thickness This Layer

□ Ticks This Layer

□ Tick Length This Layer

13. Select **Top** in the **Selection** list, check the **Show Axis & Tick** box. set **Major** as **Out** and **Minor** as **None**.



Y Axis - Layer 1

Selection: Top

☑ Show Axis & Ticks

Title:

Color: Black

Thickness(pts): 1.5

Major Tick Length: 8

Major Ticks: Out

Minor Ticks: None

Axis Position: Top

Percent/Value:

Apply To

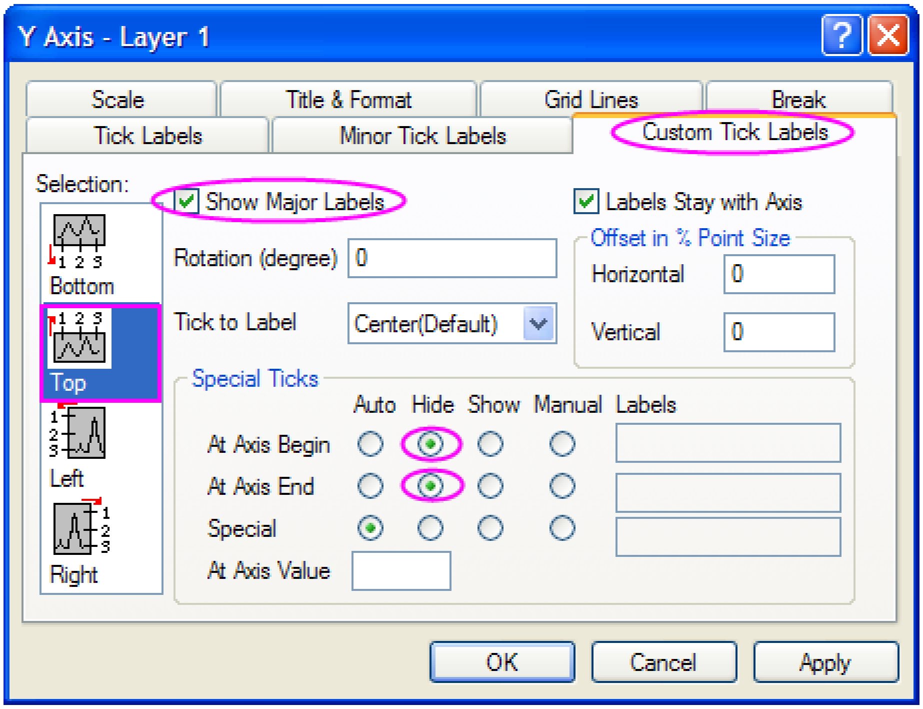
□ Color This Layer

□ Thickness This Layer

□ Ticks This Layer

□ Tick Length This Layer

14. Select the **Custom Tick Labels** tab, highlight **Top** in the Selection, choose the **Hide** radio button both with **At Axis Begin** and **At Axis End**.



Y Axis - Layer 1

Custom Tick Labels

Selection: Top

☑ Show Major Labels

☑ Labels Stay with Axis

Rotation (degree) 0

Tick to Label Center(Default)

Offset in % Point Size

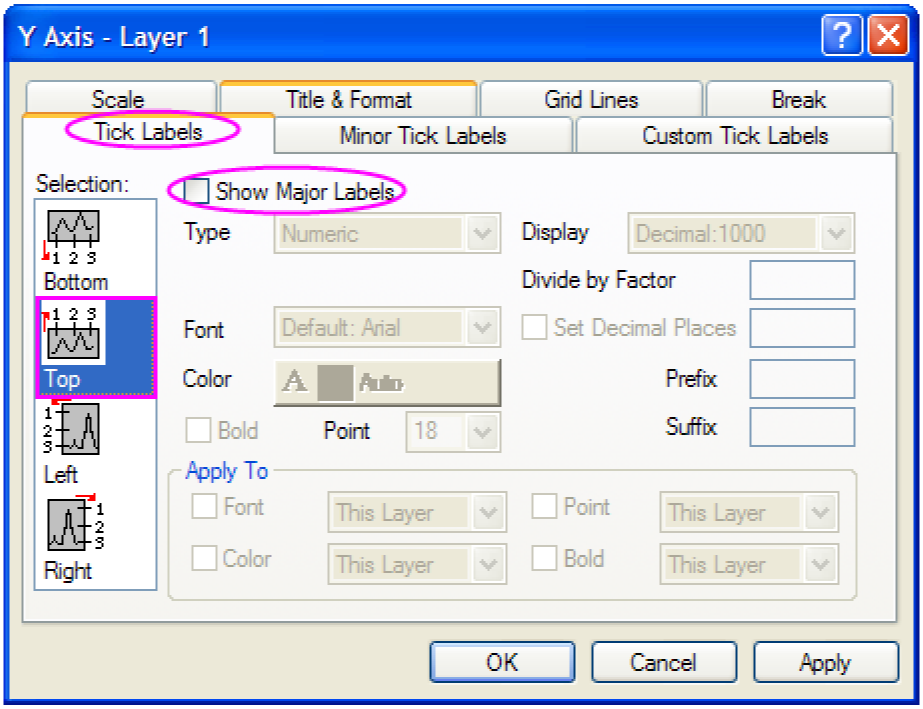
Horizontal 0

Vertical 0

Special Ticks

	Auto	Hide	Show	Manual	Labels
At Axis Begin	○	●	○	○	
At Axis End	○	●	○	○	
Special	●	○	○	○	
At Axis Value					

15. In the **Tick Labels** tab, uncheck the **Show Major Label** box.



Y Axis - Layer 1

Tick Labels

Selection: Top

□ Show Major Labels

Type Numeric

Display Decimal:1000

Divide by Factor

Font Default: Arial

□ Set Decimal Places

Color Auto

Prefix

□ Bold Point 18

Suffix

Apply To

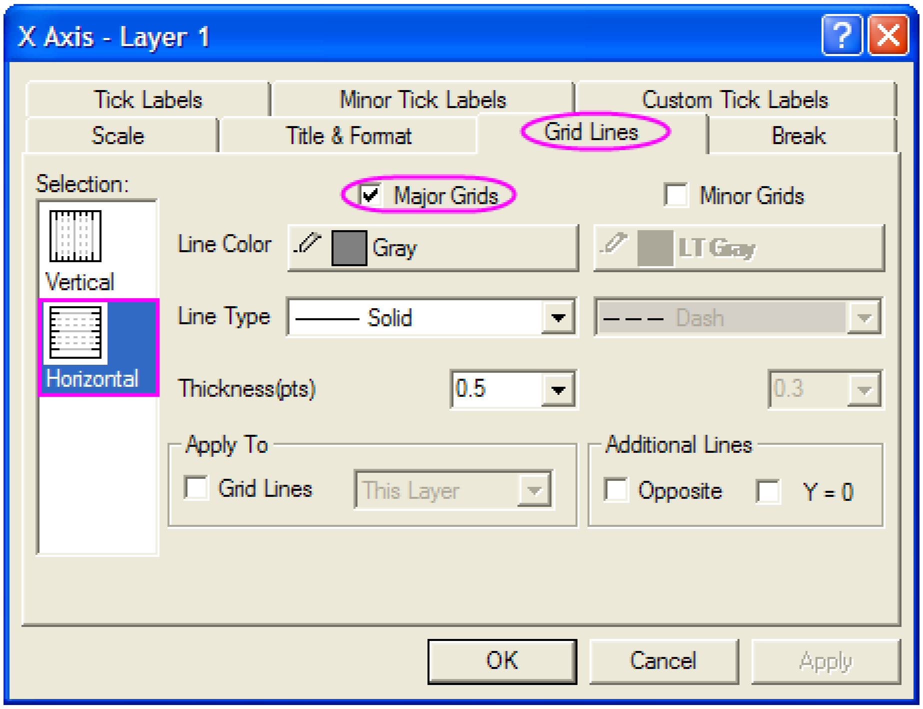
□ Font This Layer

□ Point This Layer

□ Color This Layer

□ Bold This Layer

16. In the **Grid Lines** tab, select **Horizontal** in the **Selection** list and then select the **Major Grid** checkbox. Also, choose the color and the style for the grid lines. Click the **OK** button.



Selection: Vertical, Horizontal

☑ Major Grids □ Minor Grids

Line Color: Gray | LT Gray

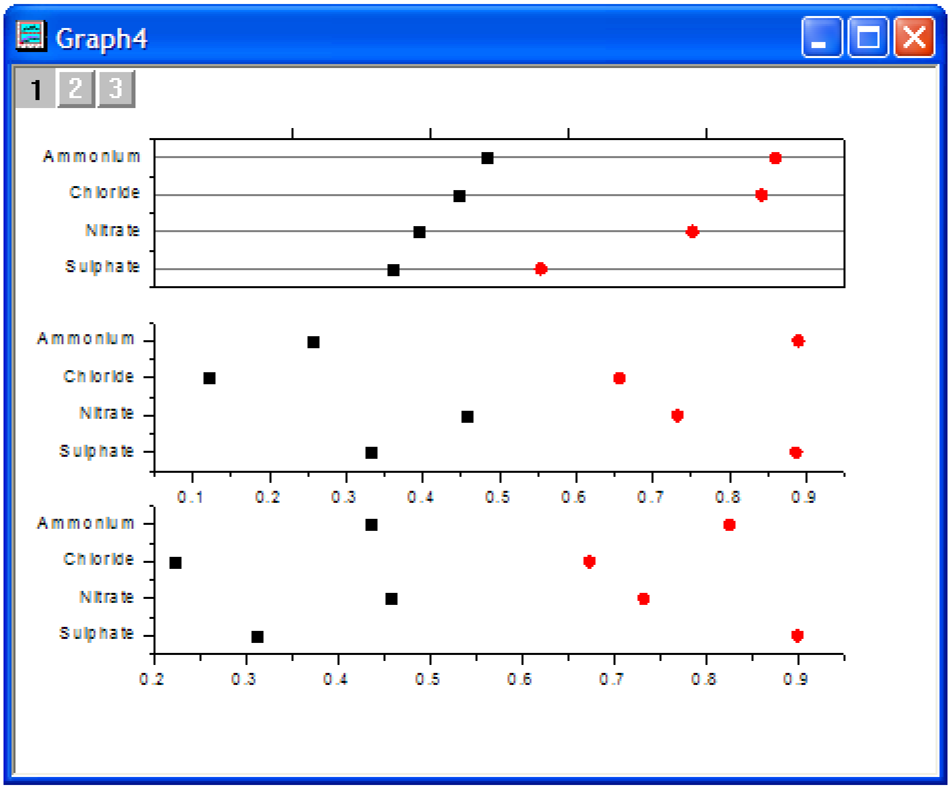
Line Type: Solid | Dash

Thickness(pts): 0.5 | 0.3

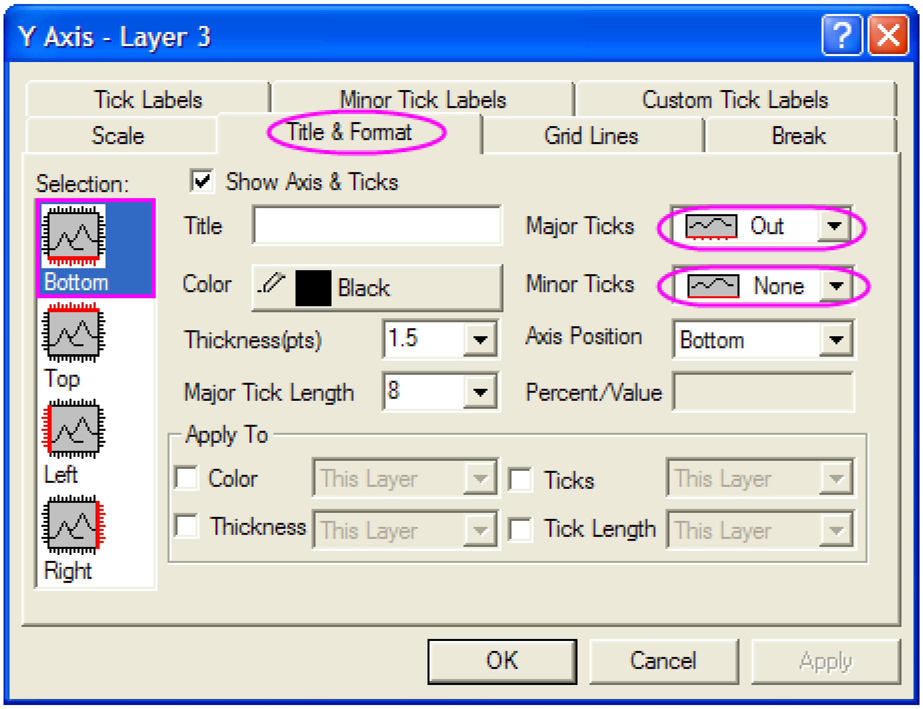
Apply To: □ Grid Lines This Layer

Additional Lines: □ Opposite □ Y = 0

17. Activate Layer 2, repeat the steps from 9 to 12 and step 16.



18. Activate Layer 3, also repeat steps from 9 to 12 and step 16 except step 10. In the **Title and Format** tab, set **Major Ticks** as **out** and **Minor Ticks** as **None**.



Selection: Bottom, Top, Left, Right (Bottom selected)

☑ Show Axis & Ticks

Title:

Color: Black

Thickness(pts): 1.5

Major Tick Length: 8

Major Ticks: Out

Minor Ticks: None

Axis Position: Bottom

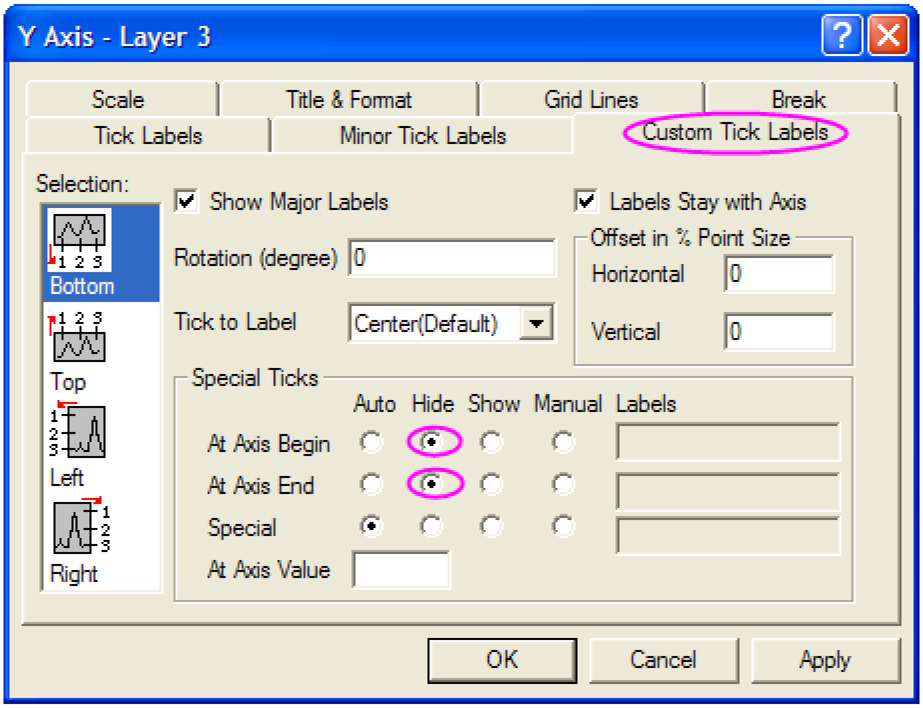
Percent/Value:

Apply To

☐ Color This Layer ☐ Ticks This Layer

☐ Thickness This Layer ☐ Tick Length This Layer

19. Select the **Custom Tick Labels** tab, highlight **Bottom** in the **Selection**, choose the **Hide** radio button both with **At Axis Begin** and **At Axis End**.



Selection: Bottom, Top, Left, Right (Bottom selected)

☑ Show Major Labels

Rotation (degree): 0

Tick to Label: Center(Default)

☑ Labels Stay with Axis

Offset in % Point Size

Horizontal: 0

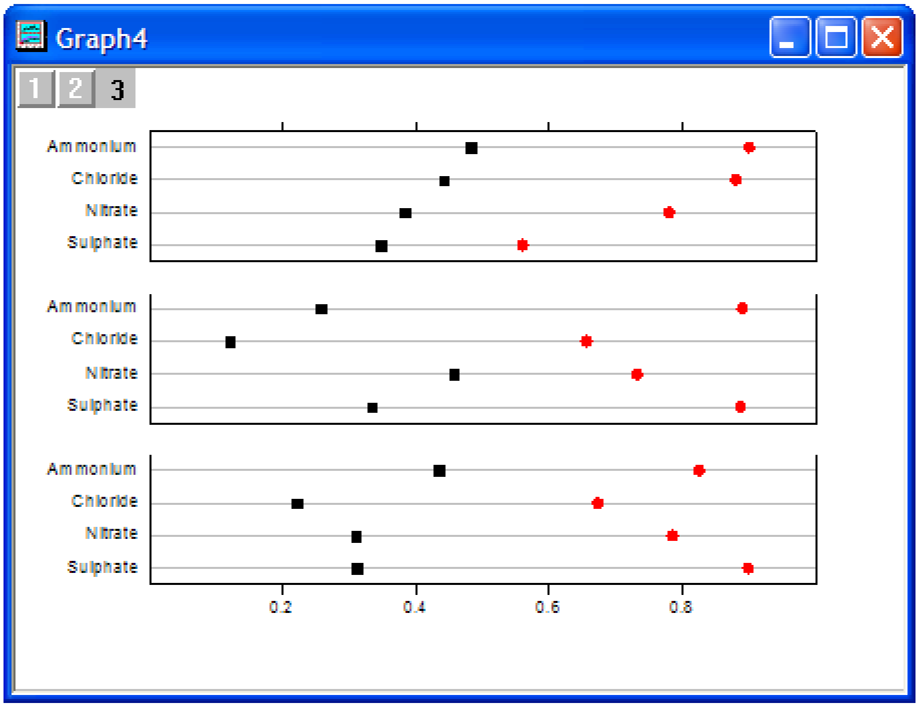
Vertical: 0

Special Ticks

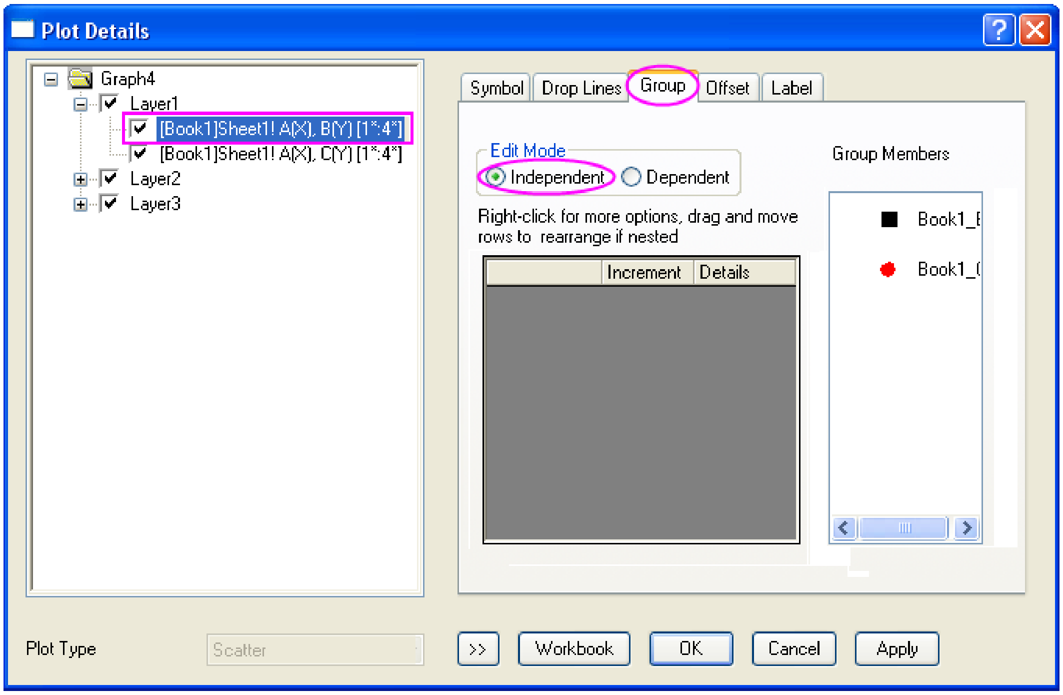
	Auto	Hide	Show	Manual	Labels
At Axis Begin	○	●	○	○	
At Axis End	○	●	○	○	
Special	●	○	○	○	

At Axis Value:

20. Click **OK**. Now you will see the graph below.



21. Double-click the graph to bring up the **Plot Details** dialog. Change the options as the following screenshot.


Plot Details

- Graph4
  - ☑ Layer1
    - ☑ [Book1]Sheet1! A(X), B(Y) [1*:4*]
    - ☑ [Book1]Sheet1! A(X), C(Y) [1*:4*]
  - ☑ Layer2
  - ☑ Layer3

Edit Mode: ● Independent ○ Dependent

Right-click for more options, drag and move rows to rearrange if nested

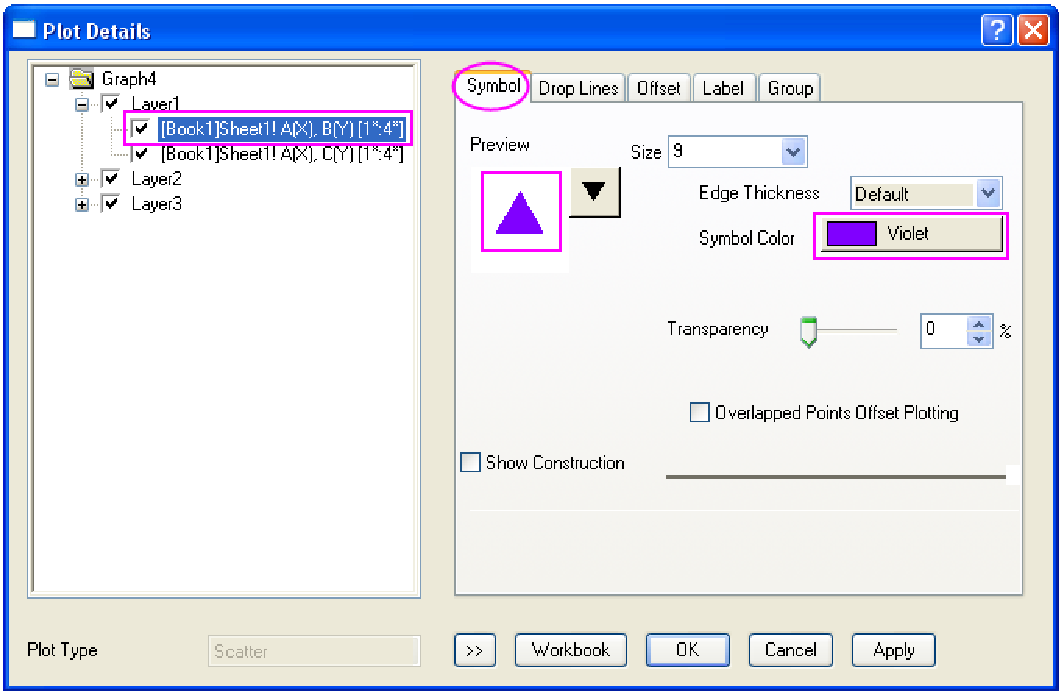
	Increment	Details

Group Members

■ Book1_E

● Book1_(

Plot Type: Scatter



Plot Details

- Graph4
  - ☑ Layer1
    - ☑ [Book1]Sheet1! A(X), B(Y) [1*:4*]
    - ☑ [Book1]Sheet1! A(X), C(Y) [1*:4*]
  - ☑ Layer2
  - ☑ Layer3

Symbol | Drop Lines | Offset | Label | Group

Preview ▲

Size: 9

Edge Thickness: Default

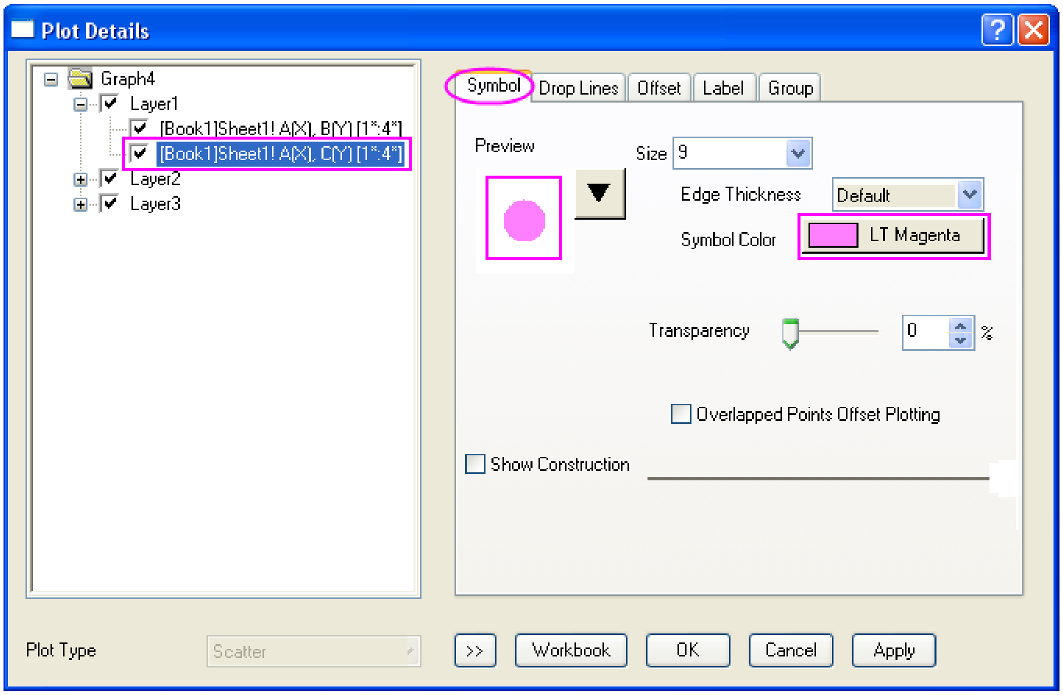
Symbol Color: Violet

Transparency: 0 %

□ Overlapped Points Offset Plotting

□ Show Construction

Plot Type: Scatter



Plot Details

- Graph4
  - ☑ Layer1
    - ☑ [Book1]Sheet1! A(X), B(Y) [1*:4*]
    - ☑ [Book1]Sheet1! A(X), C(Y) [1*:4*]
  - ☑ Layer2
  - ☑ Layer3

Symbol | Drop Lines | Offset | Label | Group

Preview ●

Size: 9

Edge Thickness: Default

Symbol Color: LT Magenta

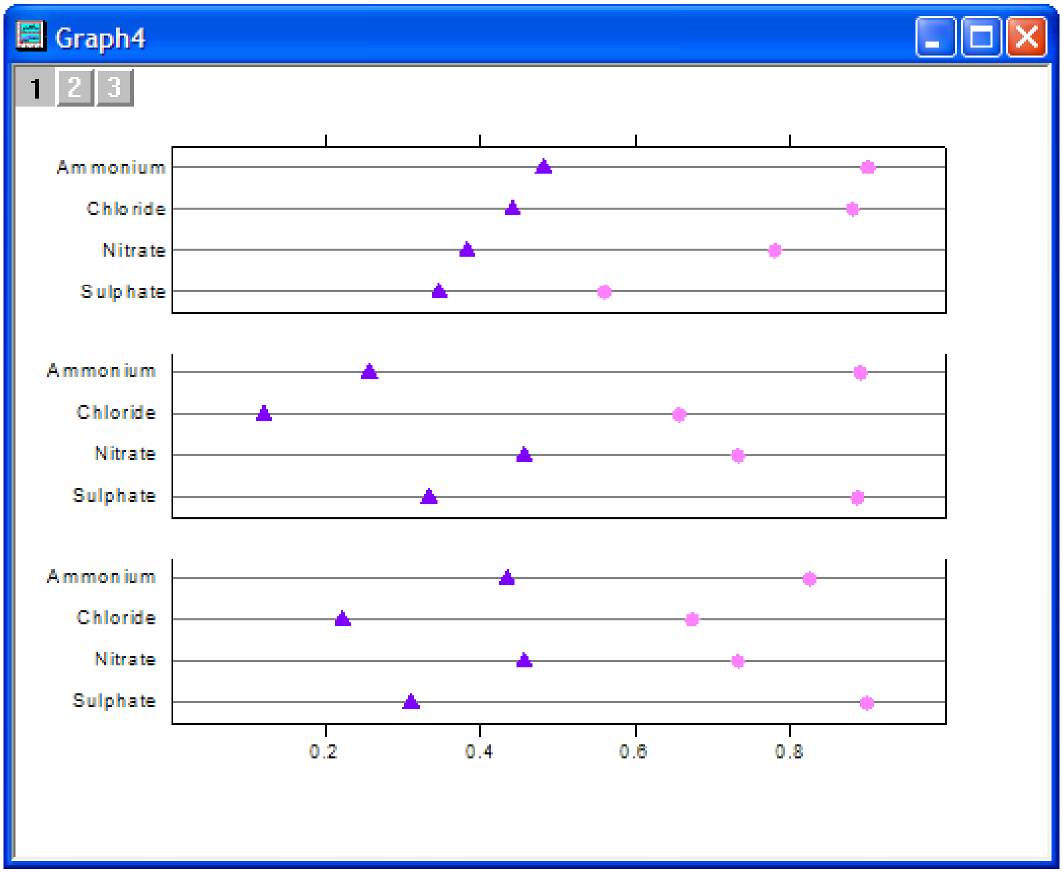
Transparency: 0 %

□ Overlapped Points Offset Plotting

□ Show Construction

Plot Type: Scatter

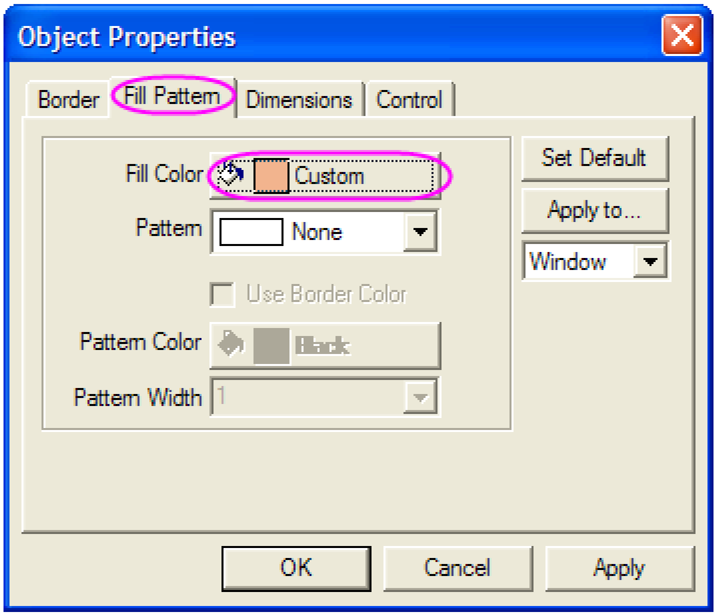
22. Repeat the steps of 21 for the Layer 2 and Layer 3. Click the **OK** button and you will see this graph.



23. Select ▭ in the **Tools** toolbar to draw three rectangles on the graphs. For each rectangle:

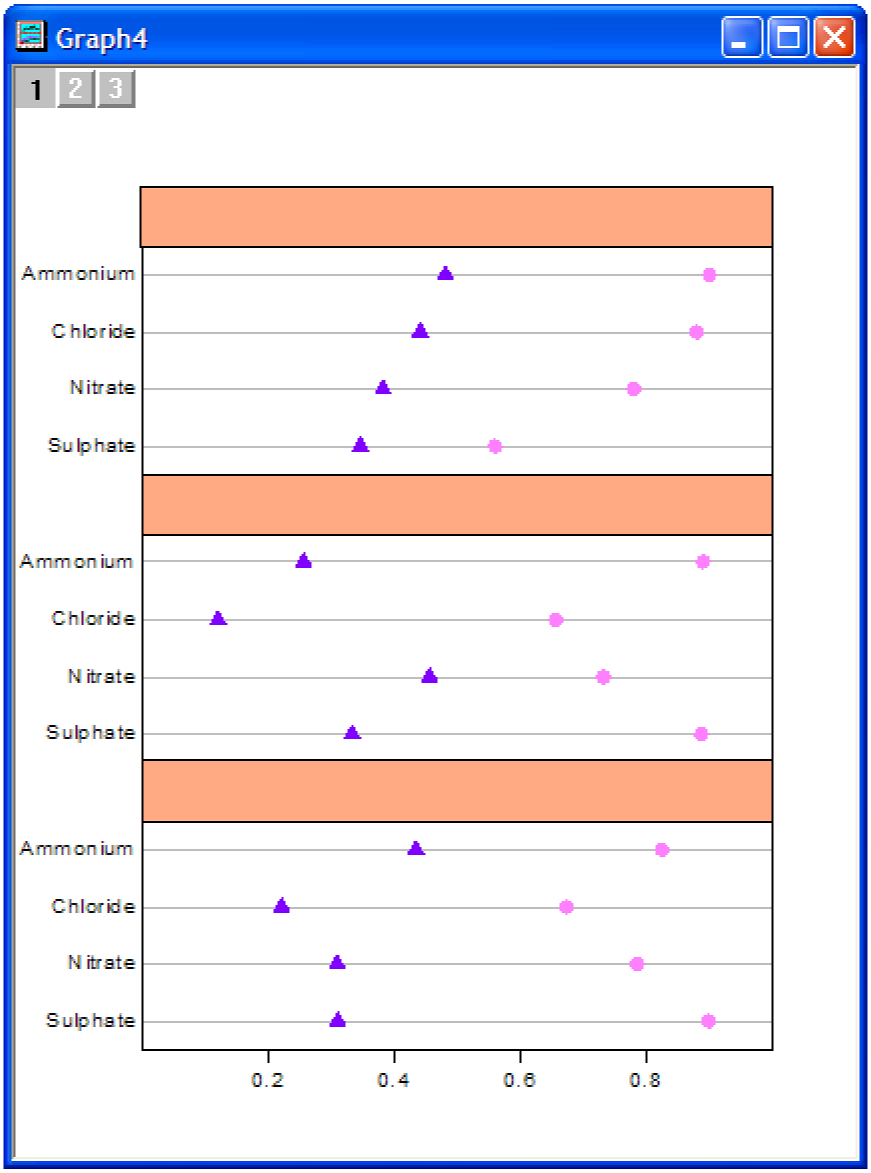
   1. Double-click on the rectangle to bring up the **Object Properties** dialog.

   2. In the **Fill Pattern** tab, set the desired **Fill Color**.



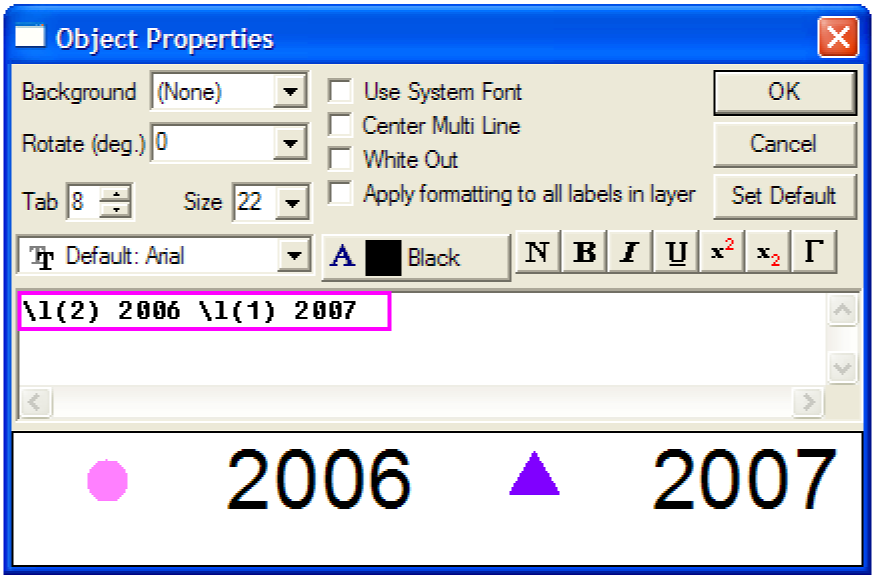
Fill Color: Custom
Pattern: None
□ Use Border Color
Pattern Color: Black
Pattern Width: 1
Window

3. In the **Dimensions** tab, adjust the size and position of the rectangle if so desired.



24. Select the Text Tool T in the Tools toolbar and click inside the rectangles to add the text that you want.

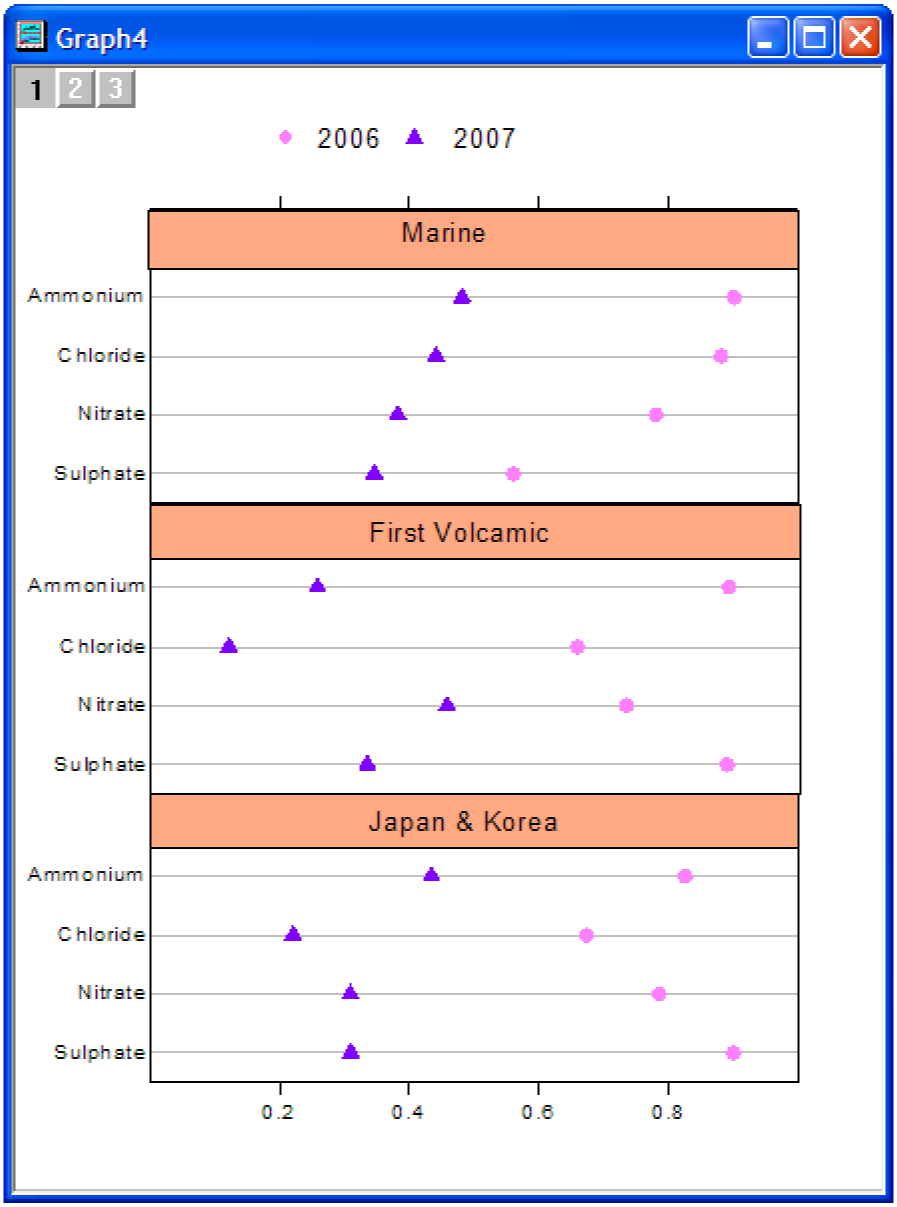
25. Select **Graph: New Legend** from the main menu. Move the legend to a suitable place, then right-click on it and select **Properties**. Change the settings as below:



```
\l(2) 2006 \l(1) 2007
```

26. Activate the top x axis of Layer 1 to move it to a suitable place.

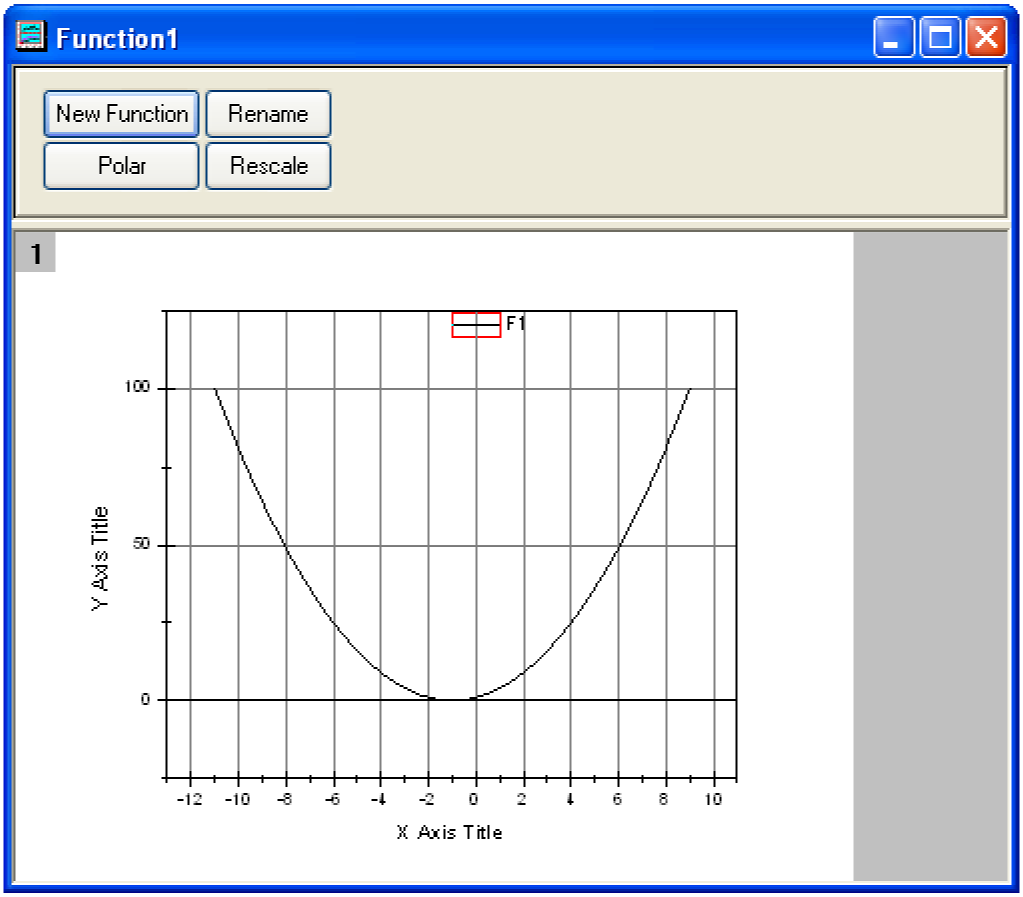
Now the multi-data dot chart is finished. You can see the graph below.

1. 

5.11 Plot Functions with Parameters Defined in a Worksheet

5.11.1 Summary

Origin can plot functions. It also can plot functions with parameters defined in a worksheet. The function graph can be updated automatically as the parameters in the worksheet change.



Minimum Origin Version Required: Origin 8.0 SR6

5.11.2 What you will learn

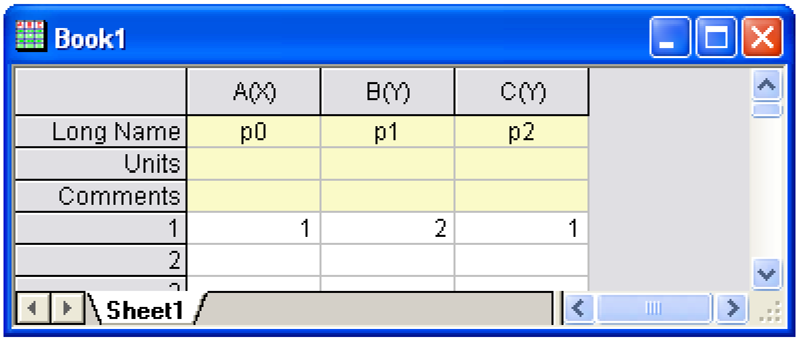
This tutorial will show you how to:

- Define variables from a worksheet in the **Set Values** dialog box.
- Plot a function graph with parameters.
- Update a graph automatically when parameters are changed.

5.11.3 Steps

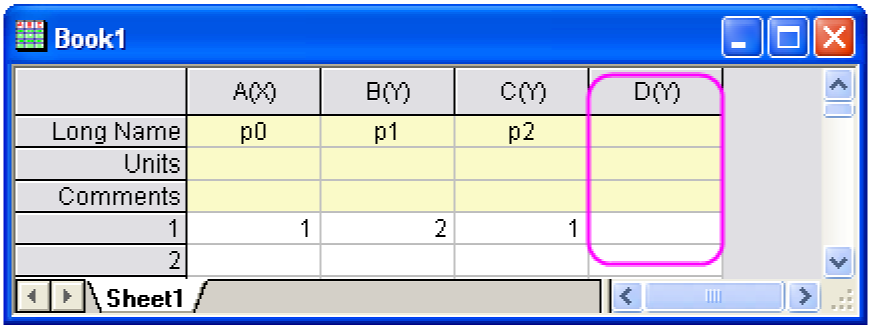
Let us use this function as an example: y=p0+p1*x+p2*x^2

1. Set up a worksheet with three parameters p0, p1, p2 stored in Column A, Column B, Column C as shown below.



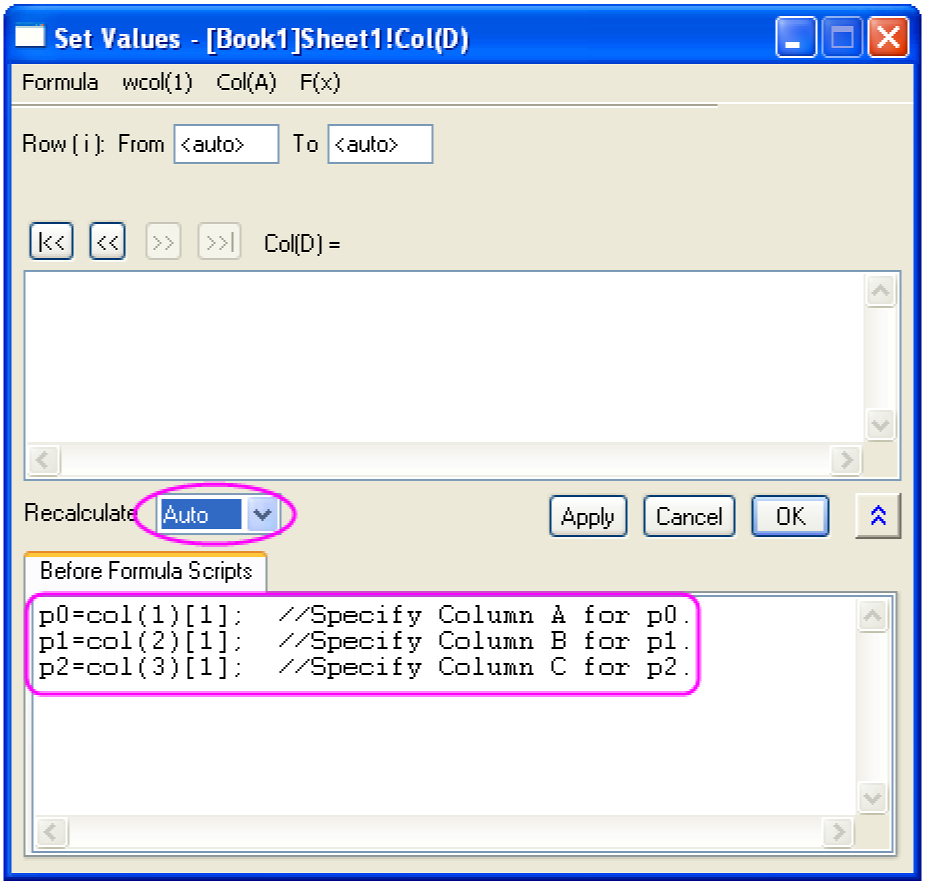
	A(X)	B(Y)	C(Y)
Long Name	p0	p1	p2
Units			
Comments			
1	1	2	1
2			

1. Click on the **Add New Columns** button on the **Standard** toolbar to add a new column to the worksheet.



	A(X)	B(Y)	C(Y)	D(Y)
Long Name	p0	p1	p2	
Units				
Comments				
1	1	2	1	
2				

2. Highlight Column D and then select **Column: Set Column Values**. Select **Auto** from the **Recalculate** drop-down. Type the script shown below to define the parameters in the **Before Formula Scripts** edit box. Click the **OK** button to close the dialog box.



Row (i): From <auto> To <auto>

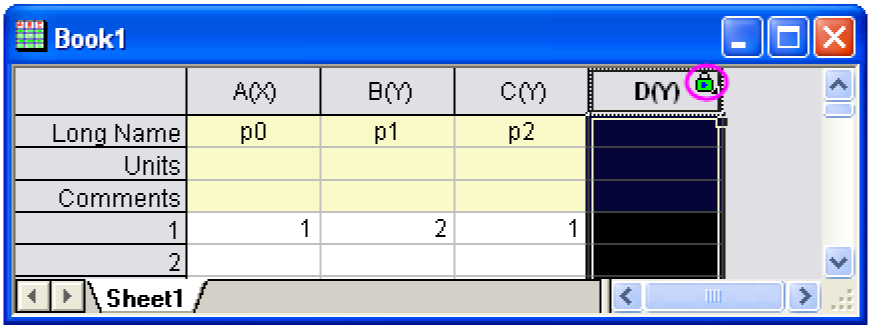
Col(D) =

Recalculate Auto

Before Formula Scripts

```
p0=col(1)[1];  //Specify Column A for p0.
p1=col(2)[1];  //Specify Column B for p1.
p2=col(3)[1];  //Specify Column C for p2.
```

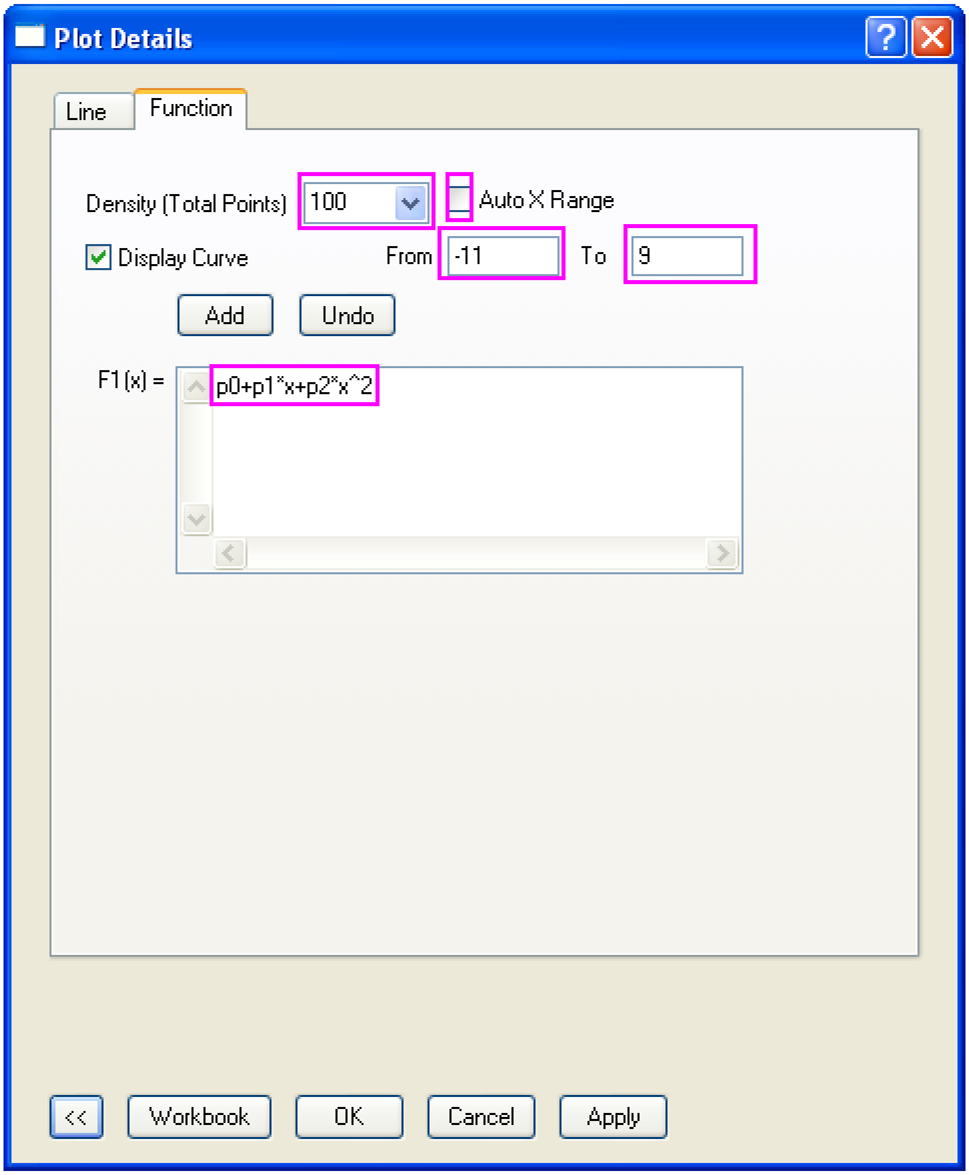
Note that there is a green lock icon on the top right corner of Column D which indicates that the Recalculate Mode is Auto.



	A(X)	B(Y)	C(Y)	D(Y)
Long Name	p0	p1	p2	
Units				
Comments				
1	1	2	1	
2				

3. Click on the **New Function** button on the **Standard** toolbar. The **Plot Details** dialog opens.

4. In the **Plot Details** dialog, set the options as follows and click the **OK** button to close the dialog box.



Density (Total Points) 100

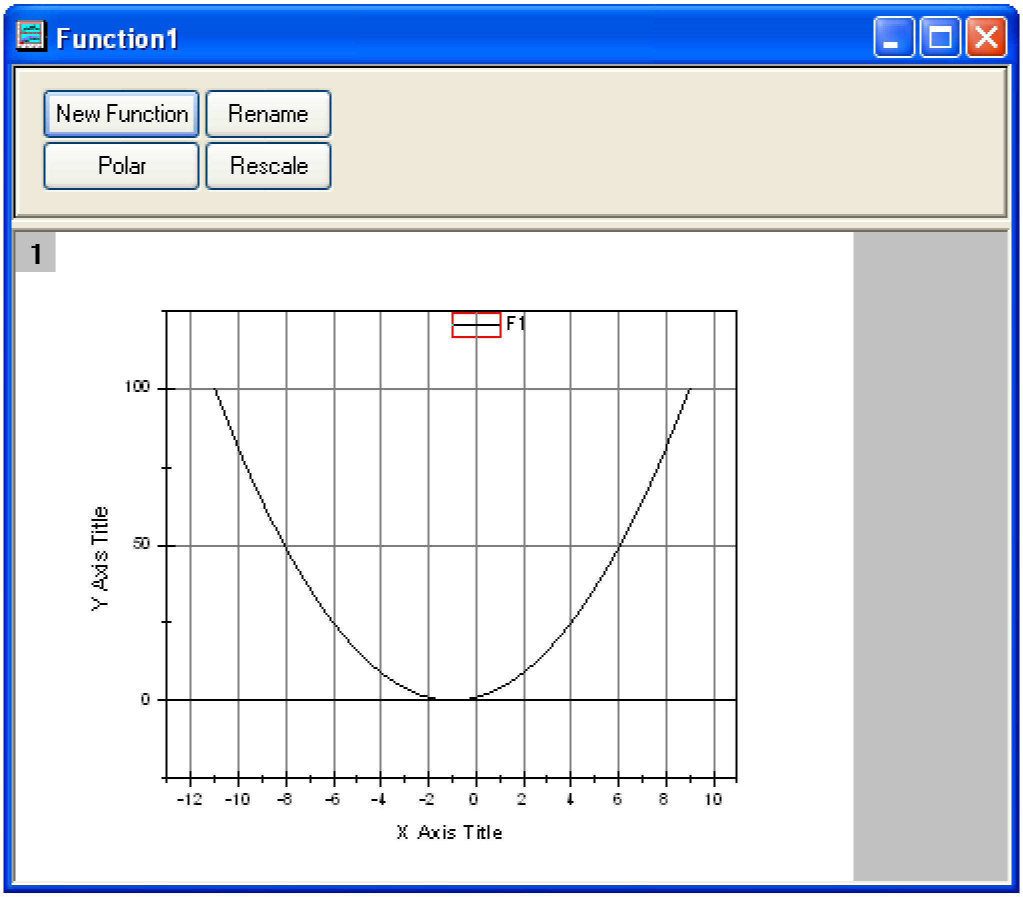
□ Auto X Range

☑ Display Curve

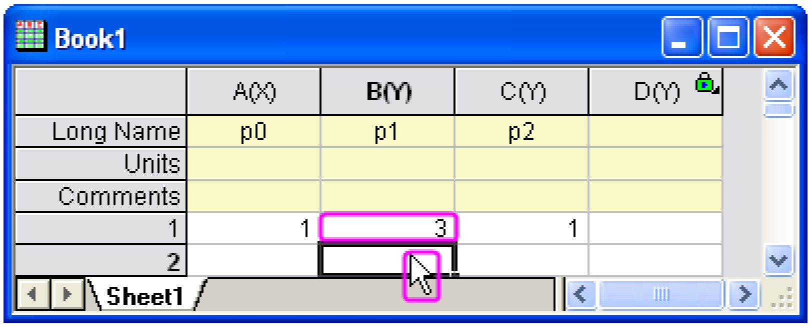
From -11 To 9

F1(x) = p0+p1*x+p2*x^2

Click on the **Rescale** button on the **Graph** toolbar to adjust the graph's scales.

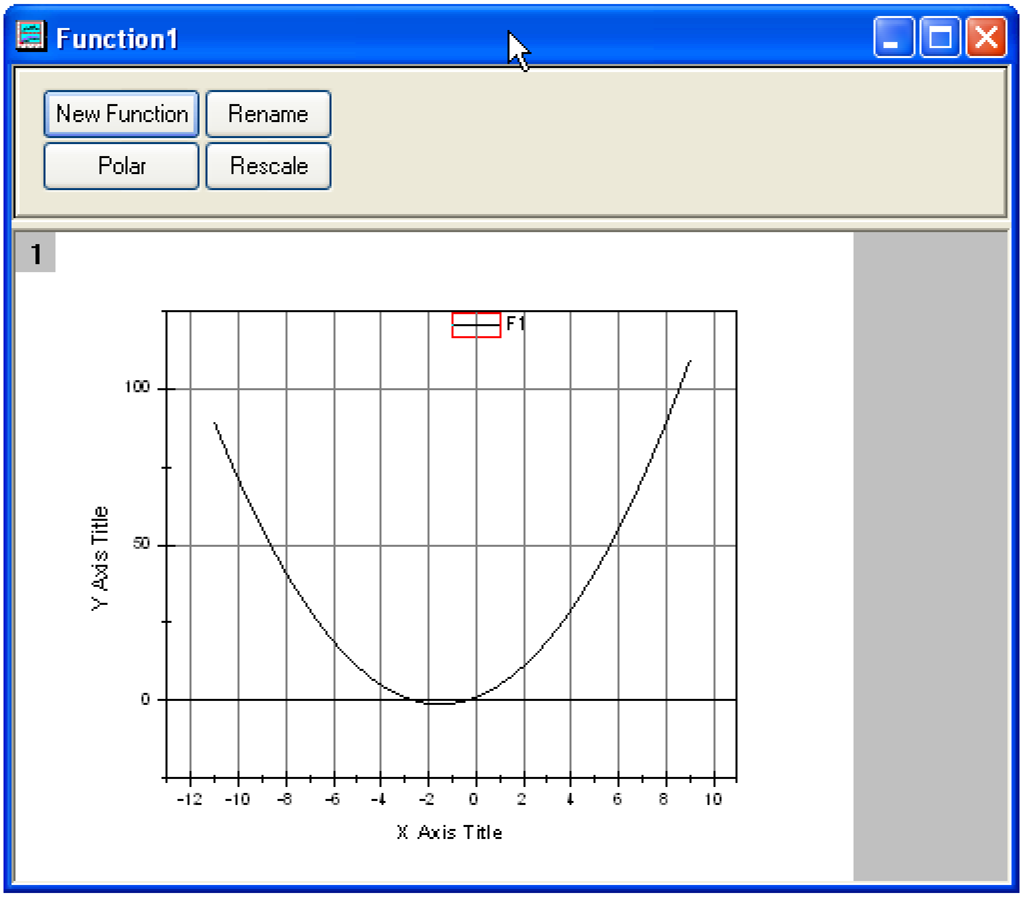


5. The function graph can be updated automatically when the parameters in the first row of the worksheet change. Change the value in the first row of Column 2 from **2** to **3**. Click outside this cell to finish editing.



	A(X)	B(Y)	C(Y)	D(Y)
Long Name	p0	p1	p2	
Units				
Comments				
1	1	3	1	
2				

The function graph updates to reflect this change.



5.11.4 Scripts

The script used in the **Before Formula Scripts** edit box of the **Set Values** dialog box is:

```
p0=col(1)[1];  //Specify Column A for p0.
p1=col(2)[1];  //Specify Column B for p1.
p2=col(3)[1];  //Specify Column C for p2.
```

The function in the **F1(x)** edit box of the **Plot Details** dialog box is as follows:

```
p0+p1*x+p2*x^2
```

6 Dialogs and Controls

6.1 Analysis Themes

6.1.1 Summary

In Origin 8, analysis procedures can be controlled by Themes. Themes are actually XML files which save settings in the analysis dialog.For example, after performing the analysis, there will now be a <Last Used> theme for this dialog which has saved the most recently used settings. You can assign a proper name for the theme and use it in the future.

For this tutorial, the Statistics on Columns dialog will be used to demonstrate how to create and use an analysis theme. This analysis provides descriptive statistics about the data such as mean, standard deviation, minimum, maximum, and more. For visualization, a histogram or box chart can also be created in the Analysis Result Sheet.

Minimum Origin Version Required: Origin 8.0 SR6

6.1.2 What you will learn

This tutorial will show you how to:

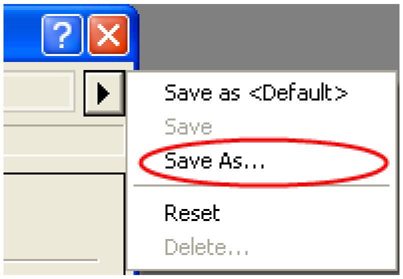
- Perform simple descriptive statistics
- Create an Analysis Theme
- Use the theme

6.1.3 Steps

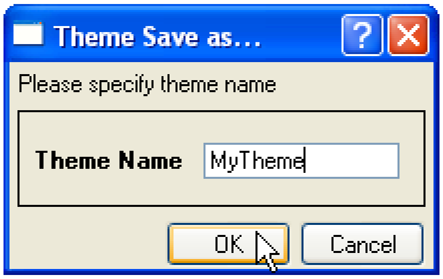
Save the analysis procedure as Theme

1. Start with a new workbook and import the file *\Samples\Statistics\automobile.dat*.
2. Highlight column C and use the menu item **Statistics: Descriptive Statistics: Statistics on Columns** to open the dialog.
3. Expand the **Moments** tree node, and check the *N Total*, *Mean*, *Standard Deviation*, *SE of Mean* and *Sum* box.
4. Expand the **Plots** tree node, and check the *Histograms* and *Box Charts* check boxes. You will then get the corresponding histogram as well as box chart graphs.

5. Your selections in this analysis dialog can be saved as your theme, so that you may easily repeat the procedure. Click the "Save Theme as..." button:



to bring up this dialog:

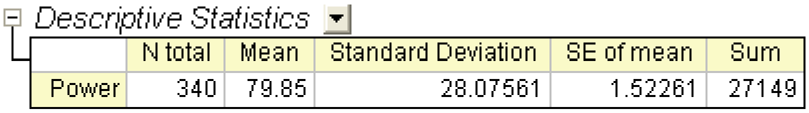

Theme Save as...

Please specify theme name

Theme Name: MyTheme

6. Type a proper theme name, such as "*MyTheme*" and click **OK** button.

7. Click the **OK** button in the **Statistics on Columns** dialog. You will see the result in a new worksheet named DescStatsOnCols1.

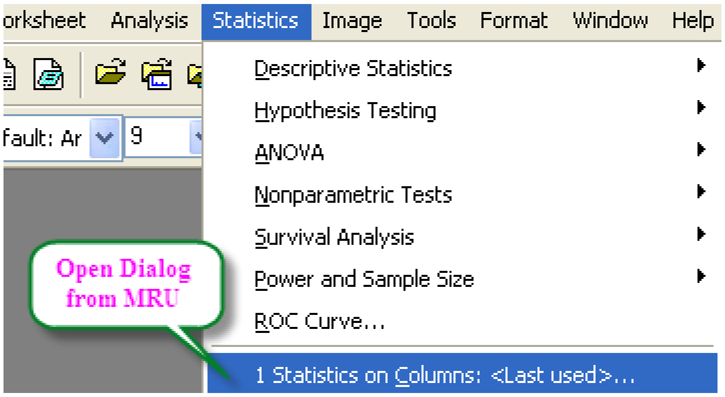

Descriptive Statistics

	N total	Mean	Standard Deviation	SE of mean	Sum
Power	340	79.85	28.07561	1.52261	27149

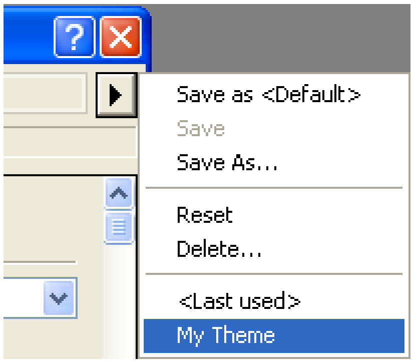
Repeat the analysis procedure by Theme

Once you save a theme, there are many ways to use it. For example, you can highlight column E and perform the same statistics on it.

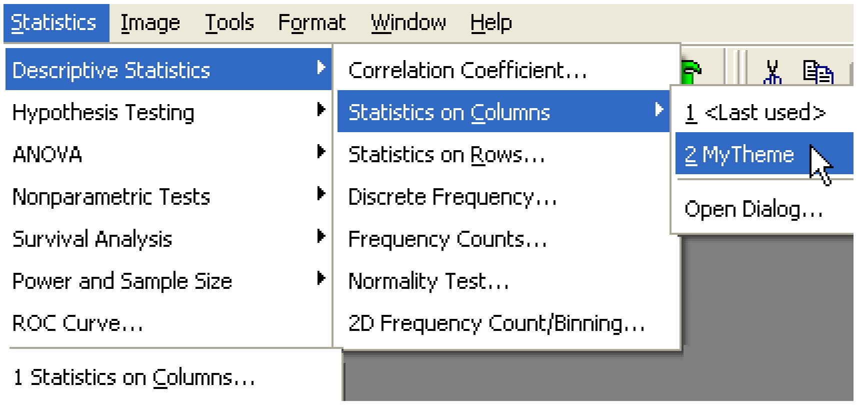
- Open the **Statistics on Columns** dialog from the **Most Recently Used** menu. Most of the menu accessible dialogs can be found from MRU.



When you open the dialog from MRU, the default theme is <Last Used>. To use the **MyTheme** theme, select *MyTheme* from the **Dialog Theme** drop-down list. The settings from that theme will then be displayed in the dialog. Click the **OK** button to do the analysis.



- Another way to apply the analysis theme is to use the cascaded menu item. Once you use an analysis dialog, or save a theme for a dialog, there will be one more menu level added. You can choose the *MyTheme* menu.



When selecting, *Open Dialog...*, the dialog will open to the <Factory Default> theme. To change the settings of your theme, you can select your theme name from the **Dialog Theme** drop-down list inside the dialog, make changes, and resave the theme. Alternatively, to open a dialog with a saved theme without performing the analysis, hold down the *shift* key while selecting the theme from the menu. This will bring up the dialog with your theme applied so you can make changes as needed.

7 Fitting

7.1 Linear Fitting and Outlier Removal

7.1.1 Summary

An outlier is typically described as a data point or observation in a collection of data points that is "very distant" from the other points and thus could be due to, for example, some fault in the measurement procedure. Identification and removal of outliers is often controversial, and is typically "more acceptable" in situations where the model used to describe the data is well known and well accepted.

Minimum Origin Version Required: Origin 8.1 SR2

7.1.2 What you will learn

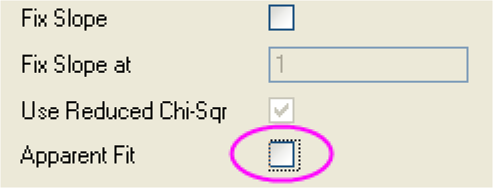
This tutorial will show you how to:

- Perform linear regression on a set of data points
- Examine the Residuals Table in the output and "identify" outliers
- Use the Masking Tool to remove the outlier points
- Use the Recalculation mechanism to automatically update the result after outlier removal

The procedure described in this tutorial is also applicable to other fitting tools such as Polynomial and Nonlinear Fitting

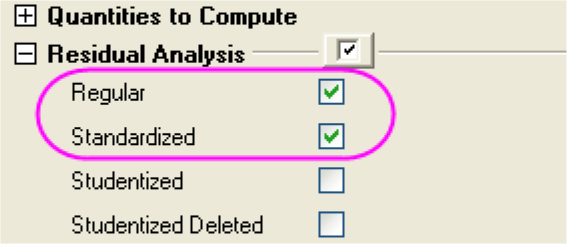
7.1.3 Steps

1. Start with a new workbook and import the file *\Samples\Curve Fitting\Outlier.dat*.
2. Click and select the second column and use the menu item **Plot: Symbol: Scatter** to create a scatter plot.
3. With the graph active, use the menu item **Analysis: Fitting: Fit Linear** to bring up the Linear Fit dialog. Note that if you have used the Linear Fit dialog before, there will be a fly-out menu and you need to select the **Open Dialog...** sub menu.
4. Under the **Fit Options** branch, clear the **Apparent Fit** check box.



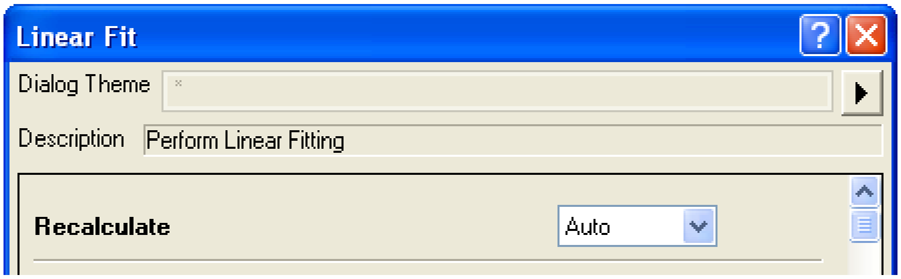
Fix Slope □
Fix Slope at 1
Use Reduced Chi-Sqr ☑
Apparent Fit □

5. Expand the **Residual Analysis** tree node in the dialog, and check the **Standardized** check box.



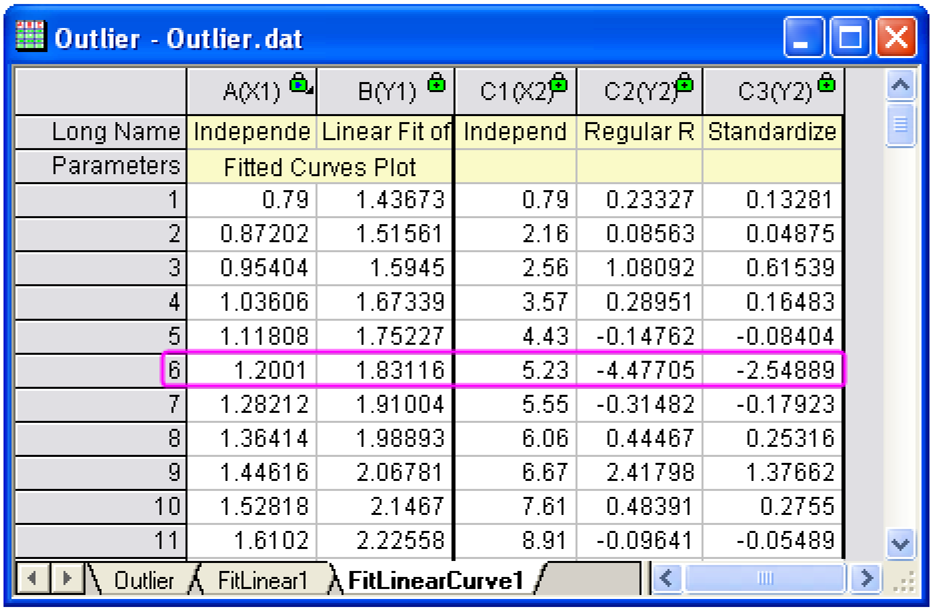
⊞ Quantities to Compute
⊟ Residual Analysis ☑
Regular ☑
Standardized ☑
Studentized □
Studentized Deleted □

6. Change the **Recalculate** drop-down at the top of the dialog to **Auto** and press the OK button at the bottom of the dialog. The dialog will close and linear regression will be performed on the data.



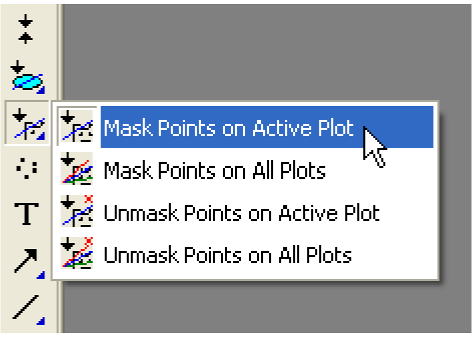
Linear Fit
Dialog Theme *
Description Perform Linear Fitting
Recalculate Auto

7. Select the **FitLinearCurve1** result sheet in the data workbook and scroll to the right side to view the **Standardized Residual** column. You will note that the value in row 6 in this columns is -2.54889:

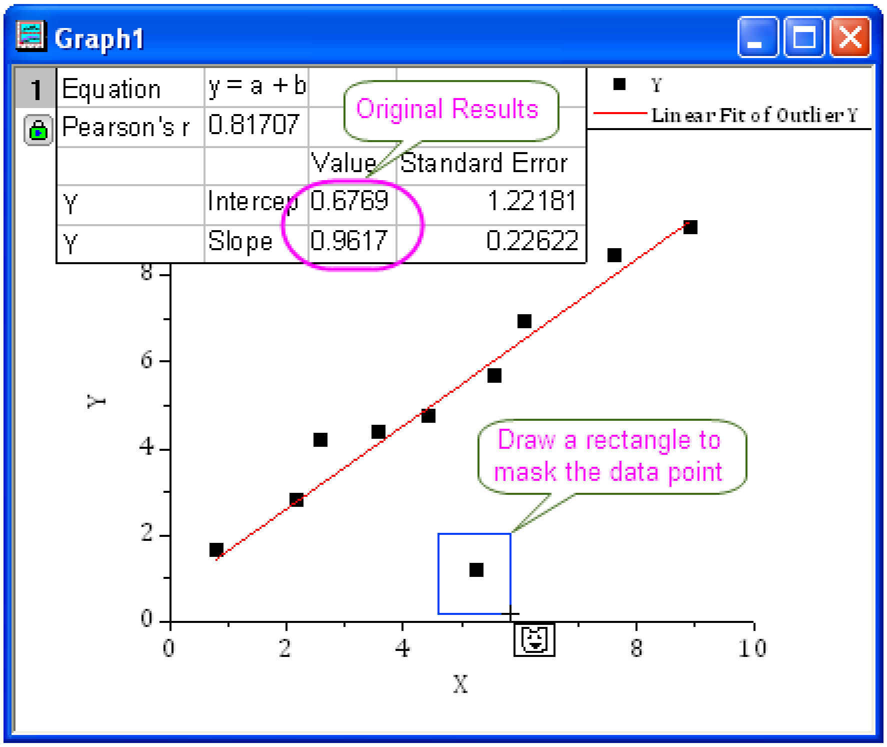


	A(X1)	B(Y1)	C1(X2)	C2(Y2)	C3(Y2)
Long Name	Independe	Linear Fit of	Independ	Regular R	Standardize
Parameters	Fitted Curves Plot				
1	0.79	1.43673	0.79	0.23327	0.13281
2	0.87202	1.51561	2.16	0.08563	0.04875
3	0.95404	1.5945	2.56	1.08092	0.61539
4	1.03606	1.67339	3.57	0.28951	0.16483
5	1.11808	1.75227	4.43	-0.14762	-0.08404
6	1.2001	1.83116	5.23	-4.47705	-2.54889
7	1.28212	1.91004	5.55	-0.31482	-0.17923
8	1.36414	1.98893	6.06	0.44467	0.25316
9	1.44616	2.06781	6.67	2.41798	1.37662
10	1.52818	2.1467	7.61	0.48391	0.2755
11	1.6102	2.22558	8.91	-0.09641	-0.05489

8. Make the graph active and then click and hold down the mouse left button on the "Regional Mask Tool" button in the Tools toolbar. Select the "Add Masked Points on Active Plot" submenu which will be the first item in the fly-out menu:


Mask Points on Active Plot
Mask Points on All Plots
Unmask Points on Active Plot
Unmask Points on All Plots
T

9. With the above submenu selected, go to the graph and click on the 6th data point to mask the point.

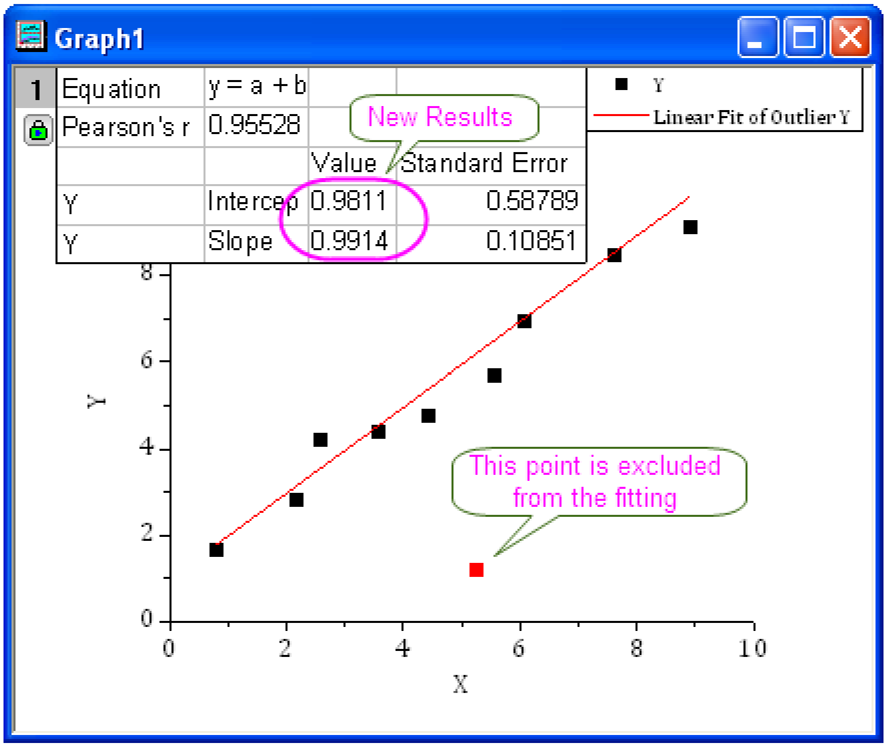


Equation	y = a + b		
Pearson's r	0.81707		
		Value	Standard Error
Y	Intercep	0.6769	1.22181
Y	Slope	0.9617	0.22622

Original Results

Draw a rectangle to mask the data point

This changes the input data to the linear fit operation and the auto update mechanism will trigger. The linear fit will be repeated with this particular masked point left out. The fit curve in the graph and the pasted parameters will automatically update. Your result graph should then look like below:



Graph1

Equation	y = a + b			
Pearson's r	0.95528			
		Value	Standard Error	
Y	Intercep	0.9811	0.58789	
Y	Slope	0.9914	0.10851	

New Results

This point is excluded from the fitting

7.2 Nonlinear Fitting with System Function

7.2.1 Summary

The NLFit dialog is an interactive tool which allows you to monitor the fitting procedure during the non-linear fitting process. This tutorial fits the Michaelis-Menten function, which is a basic model in Enzyme Kinetics, and shows you some basic features of the NLFit dialog. During the fitting, we will illustrate how to perform a Global Fit, which allows you to fit two datasets simultaneously and share some parameter values.

Minimum Origin Version Required: Origin 8.0 SR6

7.2.2 What you will learn

This tutorial will show you how to:

- Import a single ASCII file
- Perform a global fit with shared parameters
- Select a fitting range and fit part of the data
- Use the Command Window to perform simple calculation

7.2.3 Steps

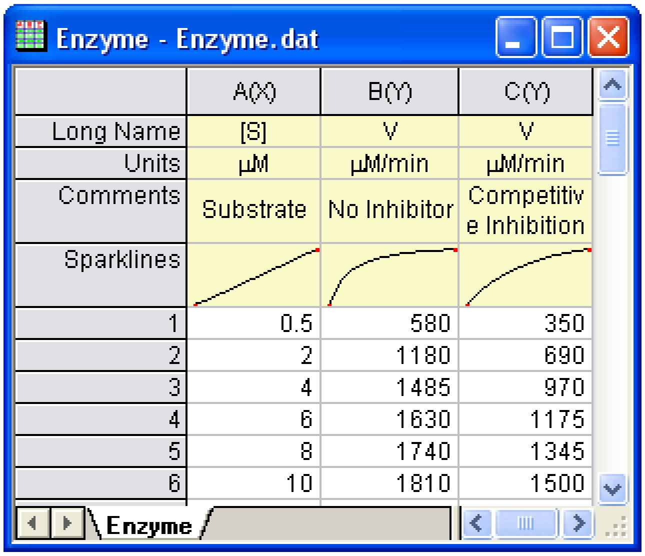
Import the file

- Open a new workbook.

- Click the *Import Single ASCII* button to bring up the **Open** dialog. Browse to *\Samples\Curve Fitting* folder and select the file *Enzyme.dat*. Make sure to check the **Show Options Dialog** checkbox at the bottom of the dialog, and then click **Open**.

- In the **impASC** dialog, expand **Import Options: Header Lines** nodes, and select **3** from *Comments From* drop down.

- Click OK to import the file.

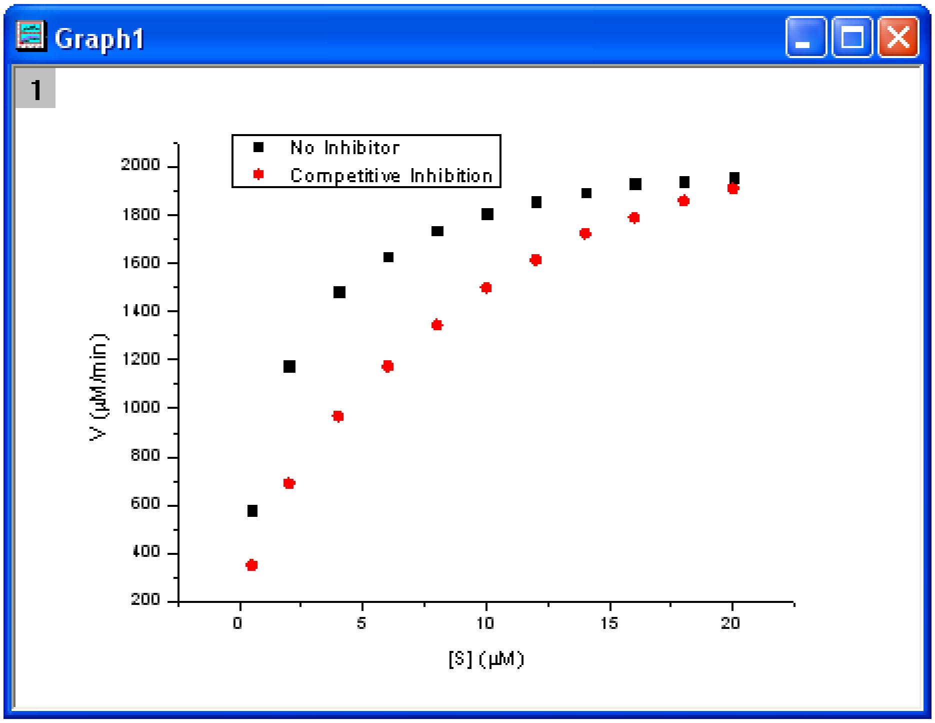


Enzyme - Enzyme.dat

	A(X)	B(Y)	C(Y)
Long Name	[S]	V	V
Units	µM	µM/min	µM/min
Comments	Substrate	No Inhibitor	Competitive Inhibition
Sparklines			
1	0.5	580	350
2	2	1180	690
3	4	1485	970
4	6	1630	1175
5	8	1740	1345
6	10	1810	1500

Plotting the Data

- Highlight columns B & C and plot as a scatter plot by clicking the button.



Fitting Michaelis-Menten Function

The single-substrate Michaelis-Menten function:

$$v = \frac{V_{max}[S]}{K_m + [S]}$$

is a basic model in enzyme kinetics study, where *v* is the reaction velocity, [*S*] is the substrate concentration, V_{max} is the maximal velocity and K_m represents the Michaelis constant. We can determine the V_{max} and K_m value, which are important enzyme properties, by fitting M-M function on *v* vs. [*S*] curve.

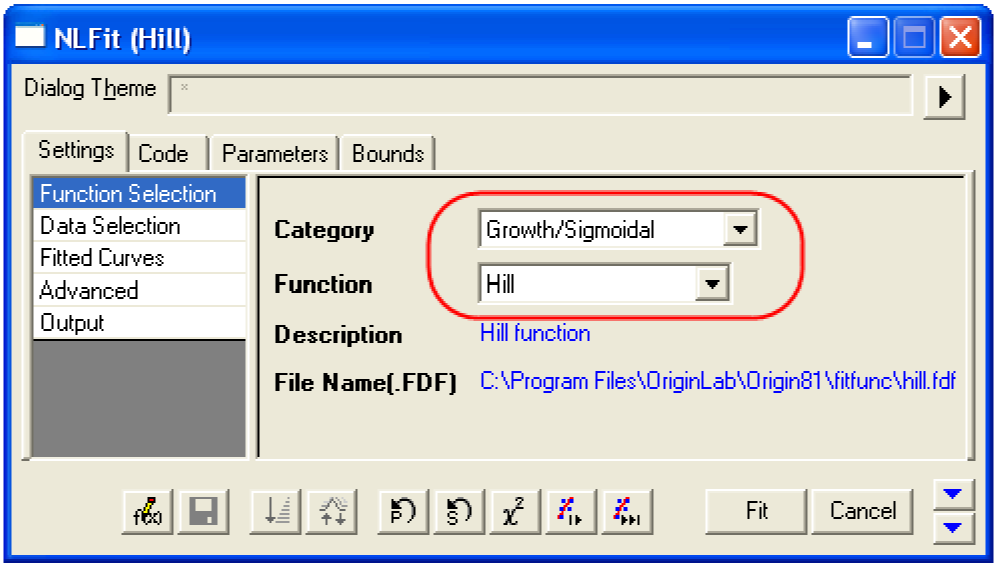
There is no M-M fitting function in Origin; however, we can use a more general model, the built-in *Hill* function to fit:

$$v = V_{max}\frac{x^n}{k^n + x^n}$$

where *n* means the cooperative sites. For single-substrate model, we can just fix *n* = 1 during fitting and it will become the simplest form, the M-M function.

There are two curves, reaction without Inhibitor and reaction with Competitive Inhibitor in the graph, and the NLFit tool can fit these two curves simultaneously. Since for competitive inhibition reaction, the maximum velocity is the same with no inhibition reaction, we can share the V_{max} value during the fitting procedure, which can be implemented by a *Global Fit*.

- With the graph active, select the menu item **Analysis: Fitting: Nonlinear Curve Fit** to bring up the NLFit dialog. Select *Hill* function from *Growth/Sigmoidal* category on the **Settings: Function Selection** page.



NLFit (Hill)

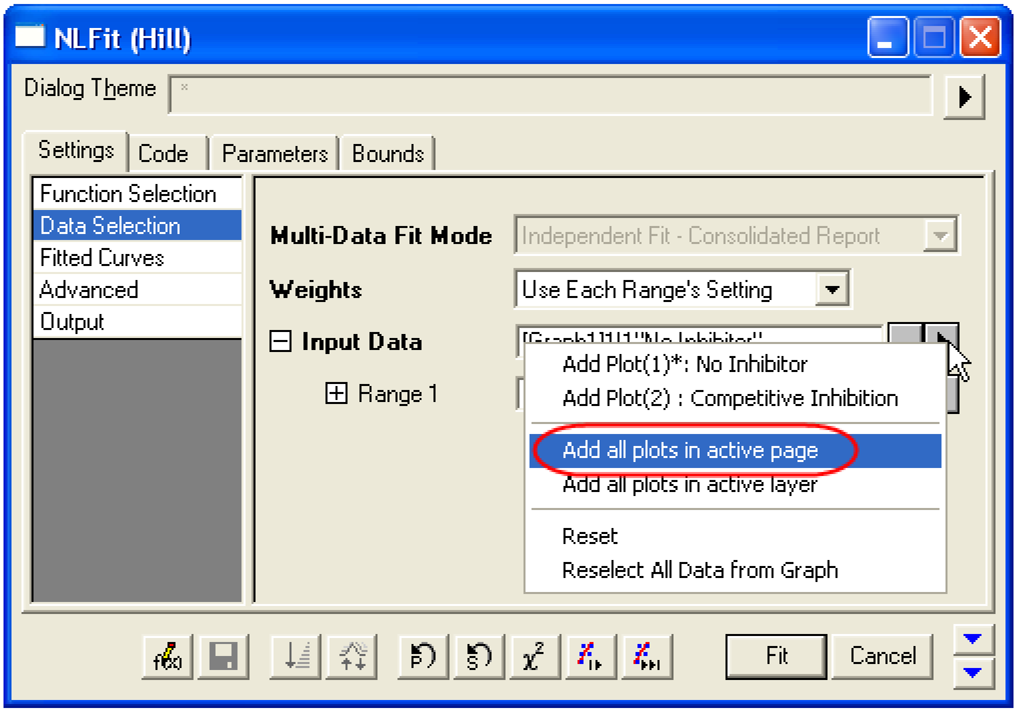
Category: Growth/Sigmoidal

Function: Hill

Description: Hill function

File Name(.FDF): C:\Program Files\OriginLab\Origin81\fitfunc\hill.fdf

- On **Settings: Data Selection** page, click the triangular button next to the **Input Data** and choose **Add all plots in active page** to set the data.



NLFit (Hill)

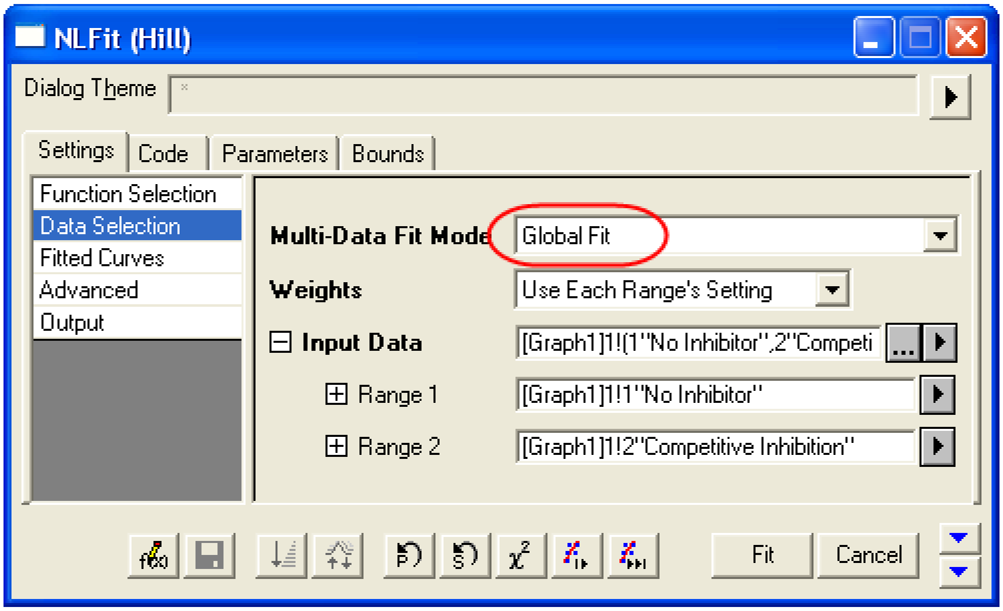
Multi-Data Fit Mode: Independent Fit - Consolidated Report

Weights: Use Each Range's Setting

Input Data: [Graph1]1!1"No Inhibitor"

Dialog Theme
Settings
Code
Parameters
Bounds
Function Selection
Data Selection
Fitted Curves
Advanced
Output
Range 1
Add Plot(1)*: No Inhibitor
Add Plot(2) : Competitive Inhibition
Add all plots in active page
Add all plots in active layer
Reset
Reselect All Data from Graph
Fit
Cancel

- Select *Global Fit* from **Multi-Data Fit Mode** drop-down list on the **Settings: Data Selection** page.



Multi-Data Fit Mode: Global Fit

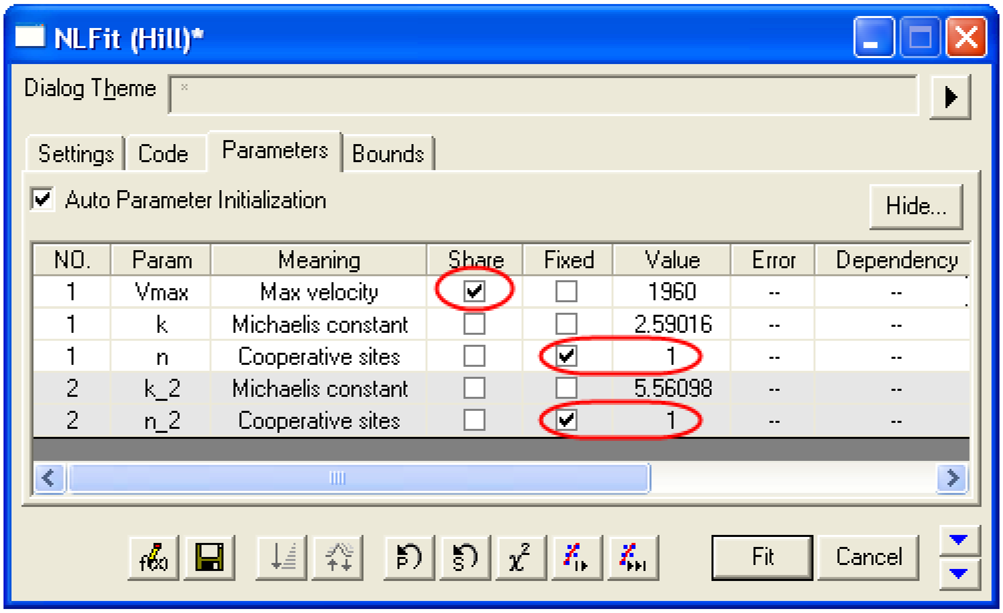
Weights: Use Each Range's Setting

Input Data: [Graph1]1!(1"No Inhibitor",2"Competi

Range 1: [Graph1]1!1"No Inhibitor"

Range 2: [Graph1]1!2"Competitive Inhibition"

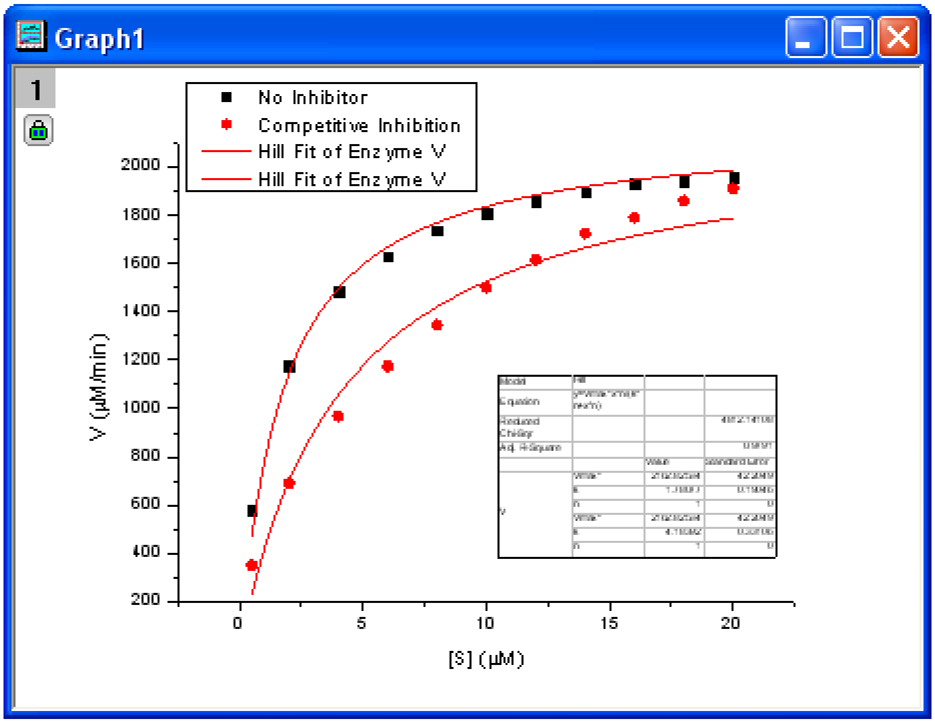
- Switch to the **Parameters** tab, check the **Share** box on the row V_{max}. These *Share* check boxes are only available when using *Global Fit* mode. Check the **Fixed** box for *n* and *n*_2, and make sure their values are 1.



☑ Auto Parameter Initialization

NO.	Param	Meaning	Share	Fixed	Value	Error	Dependency
1	Vmax	Max velocity	☑	□	1960	--	--
1	k	Michaelis constant	□	□	2.59016	--	--
1	n	Cooperative sites	□	☑	1	--	--
2	k_2	Michaelis constant	□	□	5.56098	--	--
2	n_2	Cooperative sites	□	☑	1	--	--

After that, click the *Fit* button to generate reports. The fit result will also be pasted on the original graph. (We just show the parameter values in the following figure.)



Model	Hill		
Equation	y=Vmax*x^n/(k^n+x^n)		
Reduced Chi-Sqr			[illegible]
Adj. R-Square			[illegible]
		Value	Standard Error
V	"Vmax"	[illegible]	[illegible]
	k	[illegible]	[illegible]
	n	1	0
	"Vmax"	[illegible]	[illegible]
	k	[illegible]	[illegible]
	n	1	0

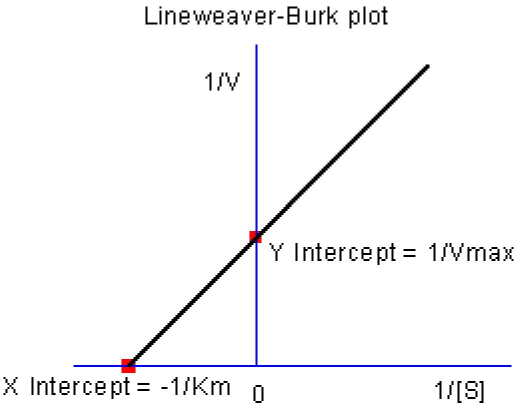
From the fit result, we can conclude that the maximum velocity is about 2160 &mu;*M* / *min*. and K_m for no inhibitor and competitive inhibitor model is 1.78&mu;*M* and 4.18&mu;*M*, respectively.

Fitting Lineweaver-Burk Plot

As we know, the model parameters can also be estimated by the Lineweaver?Burk or double-reciprocal plot. The Lineweaver?Burk plot takes the reciprocal of both sides of the M-M function and plots by $1/v$ vs. $1/[S]$:

$$\frac{1}{v}=\frac{1}{V_{max}}+\frac{K_m}{V_{max}[S]}$$

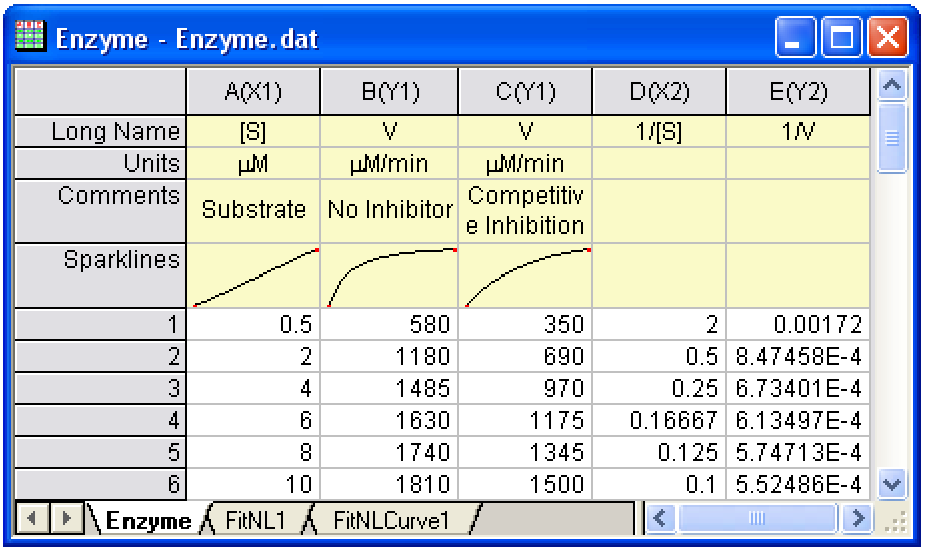
This is actually a linear function:



We will use the No Inhibitor data to illustrate how to calculate K_m and V_{max} by L-B plot.

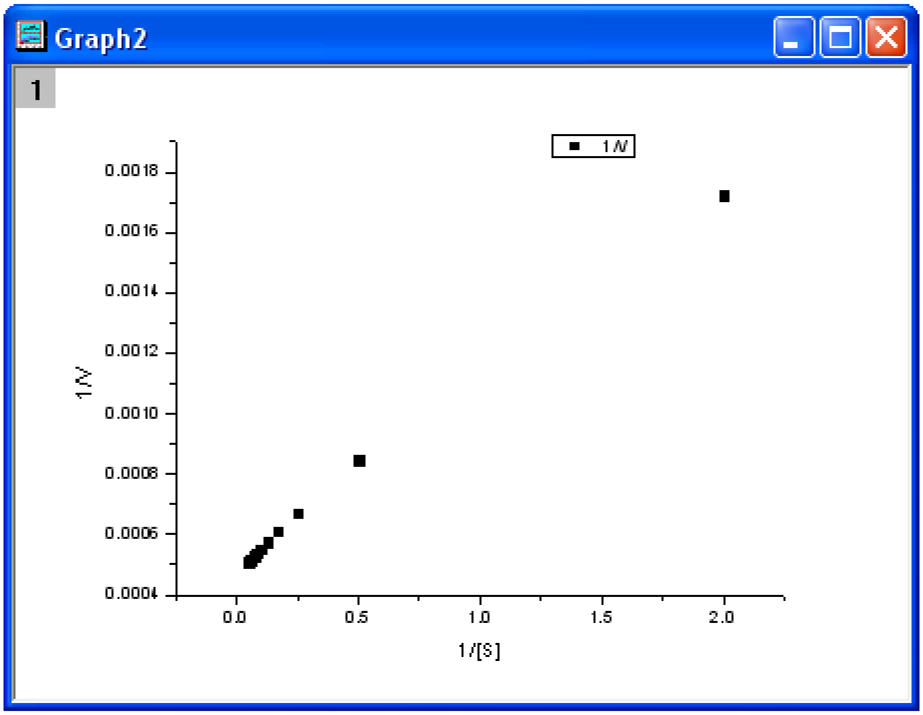
- Go back to the raw data worksheet and add two more columns by clicking the button. Right-click on column D and select **Set As: X** from the context fly-out menu to set it as an X column. Right-click on column D again and select **Set Column Values** to bring up the **Set Values** dialog. In the dialog edit box, enter: 1/Col(A) and set the **Recalculate** mode as *None*, since we don't need to auto update the reciprocal values in this example.

Similarly, set column E's values as 1/Col(B). Enter the long name for column D & E as 1 / [*S*] & 1 / *V*, respectively. And then we have:



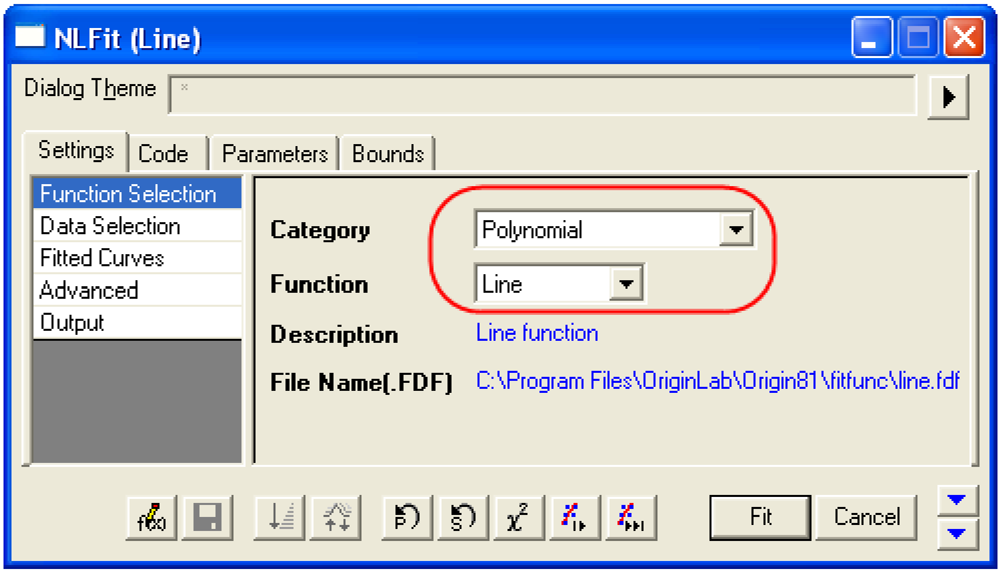
	A(X1)	B(Y1)	C(Y1)	D(X2)	E(Y2)
Long Name	[S]	V	V	1/[S]	1/V
Units	µM	µM/min	µM/min		
Comments	Substrate	No Inhibitor	Competitive Inhibition		
Sparklines					
1	0.5	580	350	2	0.00172
2	2	1180	690	0.5	8.47458E-4
3	4	1485	970	0.25	6.73401E-4
4	6	1630	1175	0.16667	6.13497E-4
5	8	1740	1345	0.125	5.74713E-4
6	10	1810	1500	0.1	5.52486E-4

- Highlight columns D & E and click button to create a scatter plot.



From the above equation, we know there is a linear relationship between 1/v and 1/[S], so we can use the NLFit tool to fit a straight line on this plot. (You can also use the Fit Linear tool from **Analysis: Fitting: Fit Linear**)

- Bring up the NLFit dialog again, select *Line* function from *Polynomial* category, and then click the *Fit* button Fit directly to generate results.



Dialog Theme *

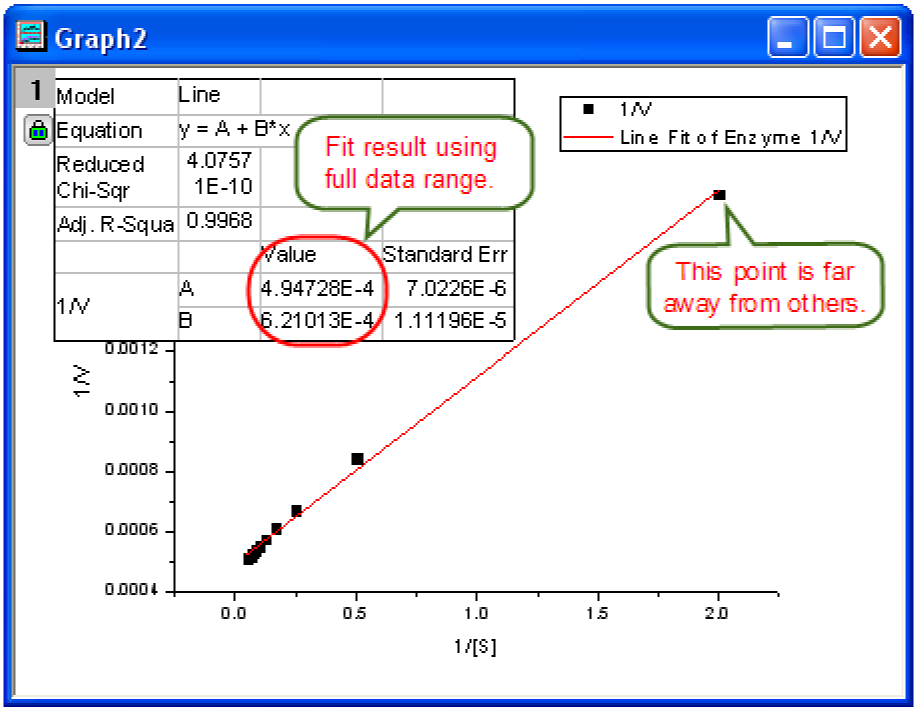
Category: Polynomial

Function: Line

Description: Line function

File Name(.FDF): C:\Program Files\OriginLab\Origin81\fitfunc\line.fdf

From the plot, one may doubt that this is the best fit curve since there is a point located far away. Actually, the right side of L-B plot is low substrate concentrations area, the measurement error may be large, so we'd better exclude these points during fitting.



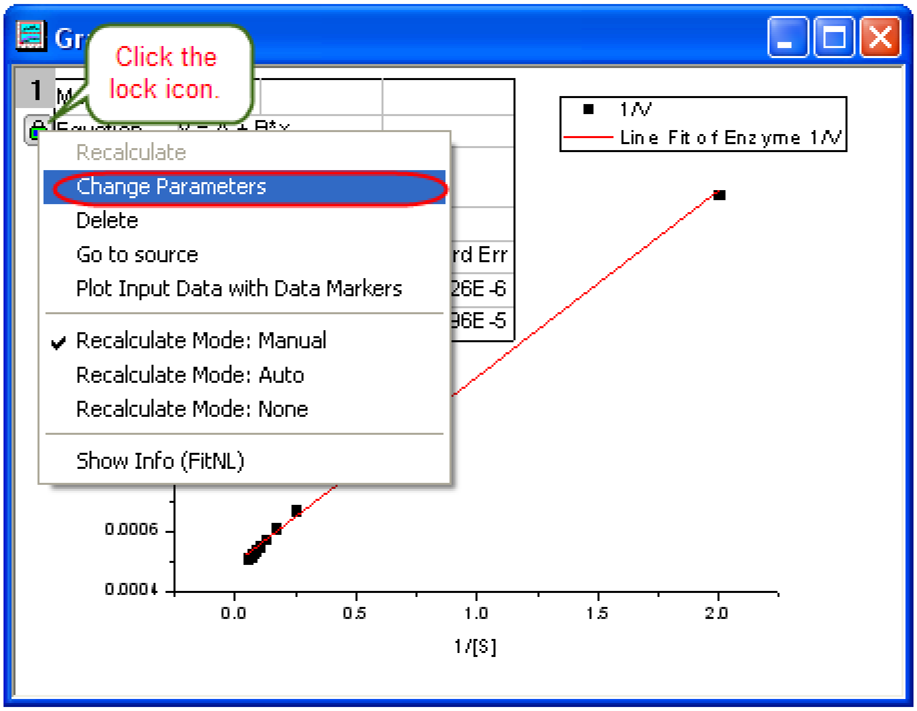
Graph2

Model	Line		
Equation	y = A + B*x		
Reduced Chi-Sqr	4.0757 1E-10		
Adj. R-Squa	0.9968		
		Value	Standard Err
1/V	A	4.94728E-4	7.0226E-6
	B	6.21013E-4	1.11196E-5

Fit result using full data range.

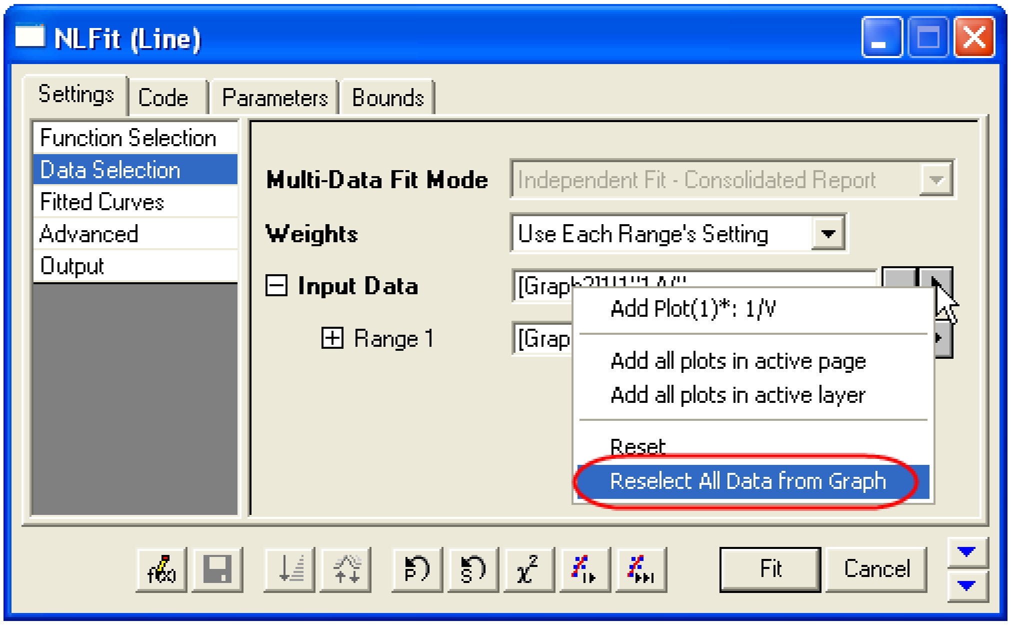
This point is far away from others.

- Click the lock icon on the graph upper-left corner, and select *Change Parameters* to bring back NLFit dialog.



Click the lock icon.

In **Settings: Data Selection** page, click the ▶button on **Input Data** node, and then choose *Reselect All Data from Graph* from fly-out menu.

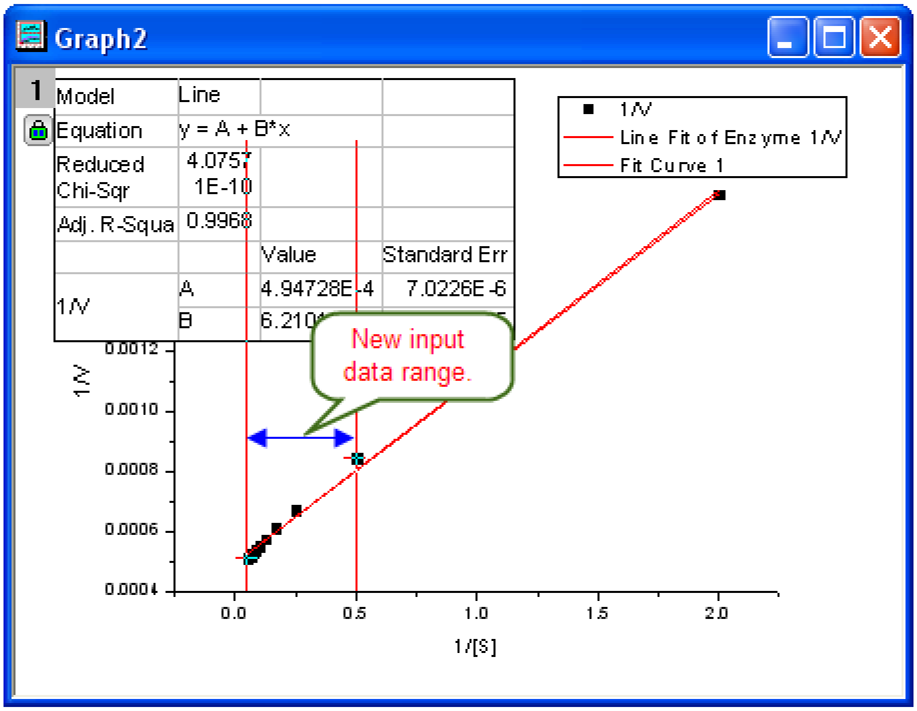


Multi-Data Fit Mode: Independent Fit - Consolidated Report

Weights: Use Each Range's Setting

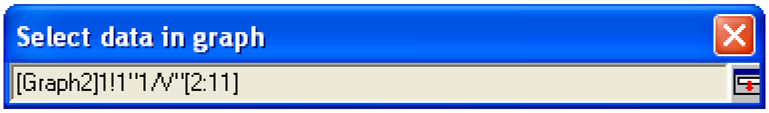
Input Data: [Graph2]1!1"1/V"

Then the NLFit dialog rolls up and your cursors become when you move to the graph page. Click and draw a rectangle to select data points you want to fit. The input range is labeled by vertical lines. You can also click-and-move these lines to change the input range.



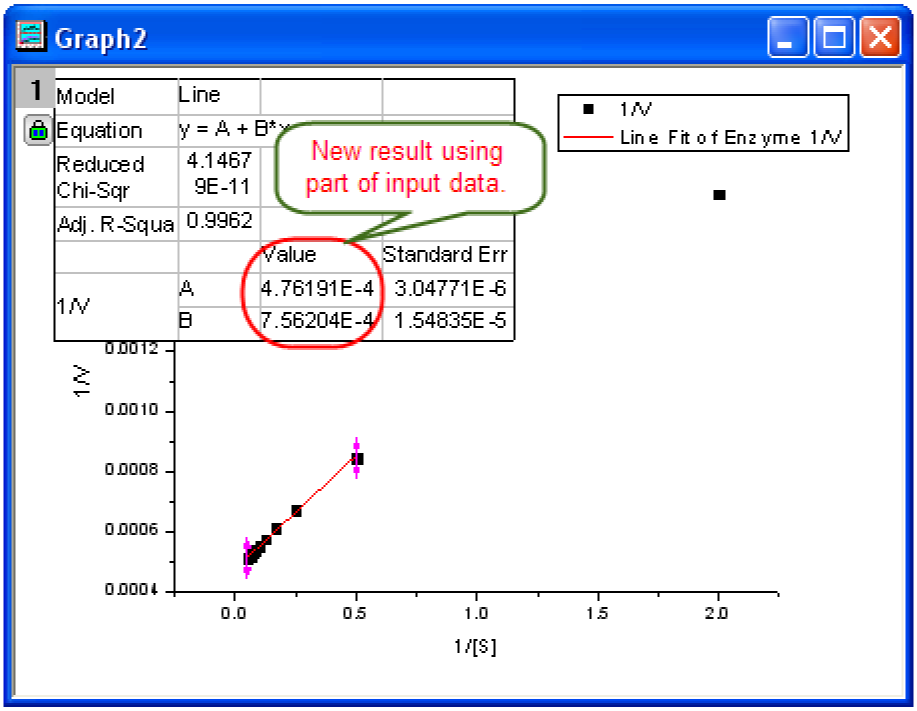
Model	Line		
Equation	y = A + B*x		
Reduced Chi-Sqr	4.0757 1E-10		
Adj. R-Squa	0.9968		
		Value	Standard Err
1/V	A	4.94728E-4	7.0226E-6
	B	6.210	E

Click the button on **Select Data in Graph** window to go back to NLFit dialog.



```
[Graph2]1!1"1/V"[2:11]
```

- Click the *Fit* button on the NLFit dialog to recalculate the result. You can see from the graph that the report table was updated.



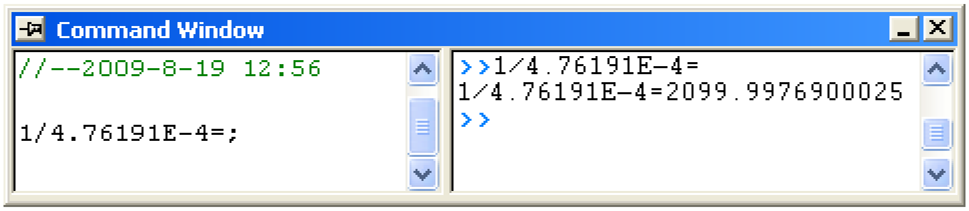
Model	Line		
Equation	y = A + B*x		
Reduced Chi-Sqr	4.1467 9E-11		
Adj. R-Squa	0.9962		
		Value	Standard Err
1/V	A	4.76191E-4	3.04771E-6
	B	7.56204E-4	1.54835E-5

New result using part of input data.

- Since the intercept of the fitted curve is 1 / V_{max}, it is equal to 4.76191E-4 in this example. To get the V_{max} value, select **Window: Command Window** to open the command window, type

```
1/4.76191E-4 =
```

and press ENTER:



```
//--2009-8-19 12:56

1/4.76191E-4=;
```

```
>>1/4.76191E-4=
1/4.76191E-4=2099.9976900025
>>
```

Origin returns the value 2099, which is close to what we got above, 2160. (When fitting the hill function above, we shared V_{max} when fitting two datasets. If you fit the No Inhibitor data only, this value will be closer.)

7.3 Global Fitting with Parameter Sharing

7.3.1 Summary

Global fit is one of the fit modes in Origin when fitting multiple curves. It will fit all datasets simultaneously, allowing parameter sharing. Compared to concatenate fit, which combine all datasets into one, global fitting performs chi-square minimization in a combined parameter space, so the parameter errors, DOF, npts and even parameter values may be different from a concatenated fit. Therefore global fitting is only appropriate/necessary if you want to share parameters.

Minimum Origin Version Required: Origin 8.0 SR6

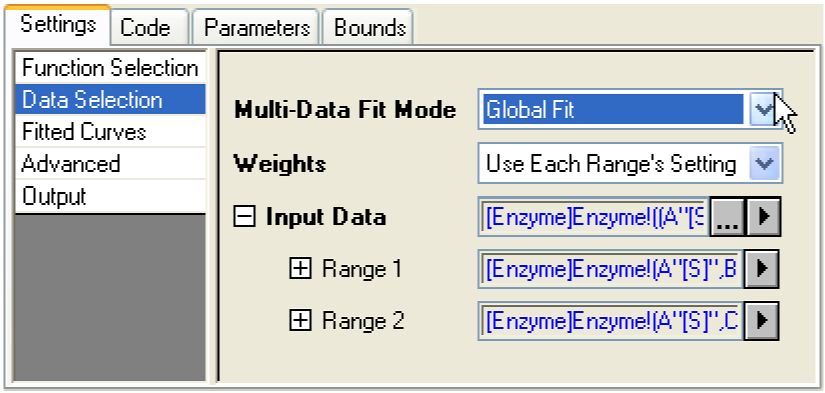
7.3.2 What you will learn

This tutorial will show you how to:

- Select multiple datasets for fitting.
- Select different fitting modes.
- Perform global fit with parameter sharing.

7.3.3 Steps

1. Start with a new workbook and import the file *\Samples\Curve Fitting\Enzyme.dat*.
2. Highlight column B and C and bring up the **NLFit** dialog from **Analysis: Fitting: Nonlinear Curve Fitting**. In the **Function Selection** page of **NLFit** dialog, choose **Hill** function from **Growth/Sigmoidal** category. Go to **Data Selection** page, and select **Global Fit** mode from **Multi-Data Fit Mode** drop-down list:



Multi-Data Fit Mode Global Fit

Weights Use Each Range's Setting

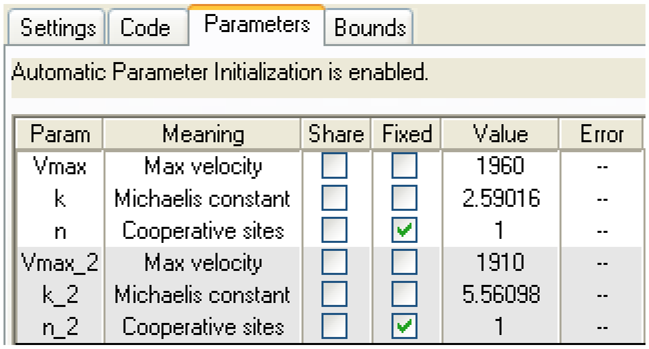
⊟ **Input Data** [Enzyme]Enzyme!((A"[S

⊞ Range 1 [Enzyme]Enzyme!(A"[S]",B

⊞ Range 2 [Enzyme]Enzyme!(A"[S]",C

Then make sure the **Recalculate** mode is **Manual** in the **Advanced** page.

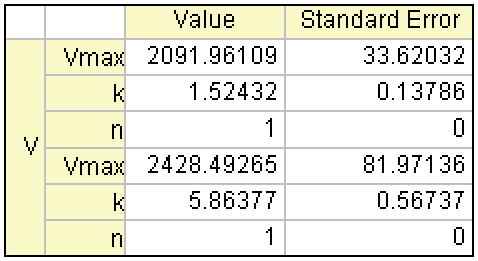
3. Active the **Parameters** tab. Check the **Fixed** checkbox for *n* and *n_2* to fix their values to 1.



Automatic Parameter Initialization is enabled.

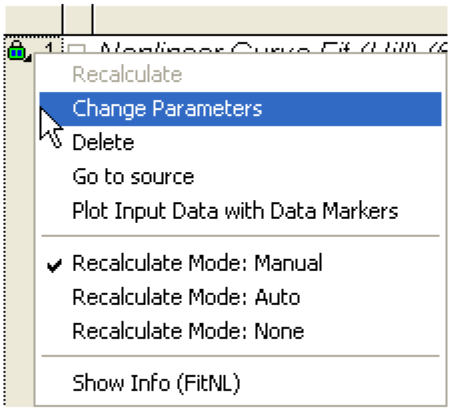
Param	Meaning	Share	Fixed	Value	Error
Vmax	Max velocity	☐	☐	1960	--
k	Michaelis constant	☐	☐	2.59016	--
n	Cooperative sites	☐	☑	1	--
Vmax_2	Max velocity	☐	☐	1910	--
k_2	Michaelis constant	☐	☐	5.56098	--
n_2	Cooperative sites	☐	☑	1	--

Click the **Fit** button to fit curves. You can see these results from the report worksheet:

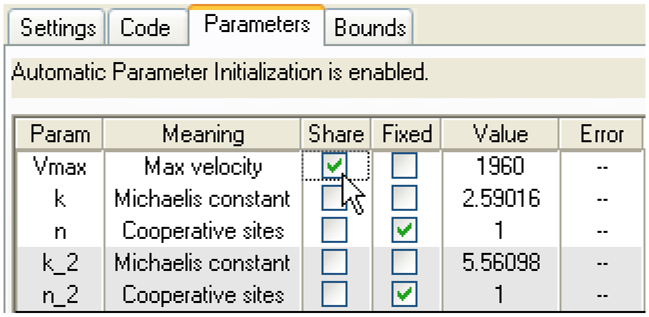


		Value	Standard Error
V	Vmax	2091.96109	33.62032
	k	1.52432	0.13786
	n	1	0
	Vmax	2428.49265	81.97136
	k	5.86377	0.56737
	n	1	0

4. Since the maximum velocity, *Vmax* in this case, maybe the same. We now want to share this parameter value to fit. Click the lock icon in the report worksheet and select **Change Parameters** to bring back the **NLFit** dialog.



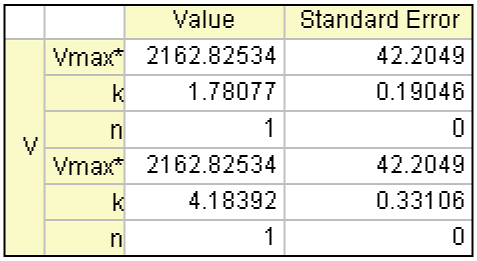
5. In the **Parameters** tab, check the **Shared** checkbox for *Vmax*.



Automatic Parameter Initialization is enabled.

Param	Meaning	Share	Fixed	Value	Error
Vmax	Max velocity	☑	□	1960	--
k	Michaelis constant	□	□	2.59016	--
n	Cooperative sites	□	☑	1	--
k_2	Michaelis constant	□	□	5.56098	--
n_2	Cooperative sites	□	☑	1	--

And then click the **Fit** button again to generate new results, you can see the *Vmax* values for both curves are the same. The asterisk in parameter name means that this parameter is shared:



		Value	Standard Error
V	Vmax*	2162.82534	42.2049
	k	1.78077	0.19046
	n	1	0
	Vmax*	2162.82534	42.2049
	k	4.18392	0.33106
	n	1	0

7.4 User Defined Fitting Function using Origin C

7.4.1 Summary

All the fitting functions in Origin are organized by **Fitting Function Organizer**. Beside the build-in functions, you can also create user-define functions in function organizer. Once a function is created, it can be accessed in the NLFit dialog. We will illustrate how to fit by user-define function below.

Minimum Origin Version Required: Origin 8.0 SR6

7.4.2 What you will learn

- Create a user-define fitting function.

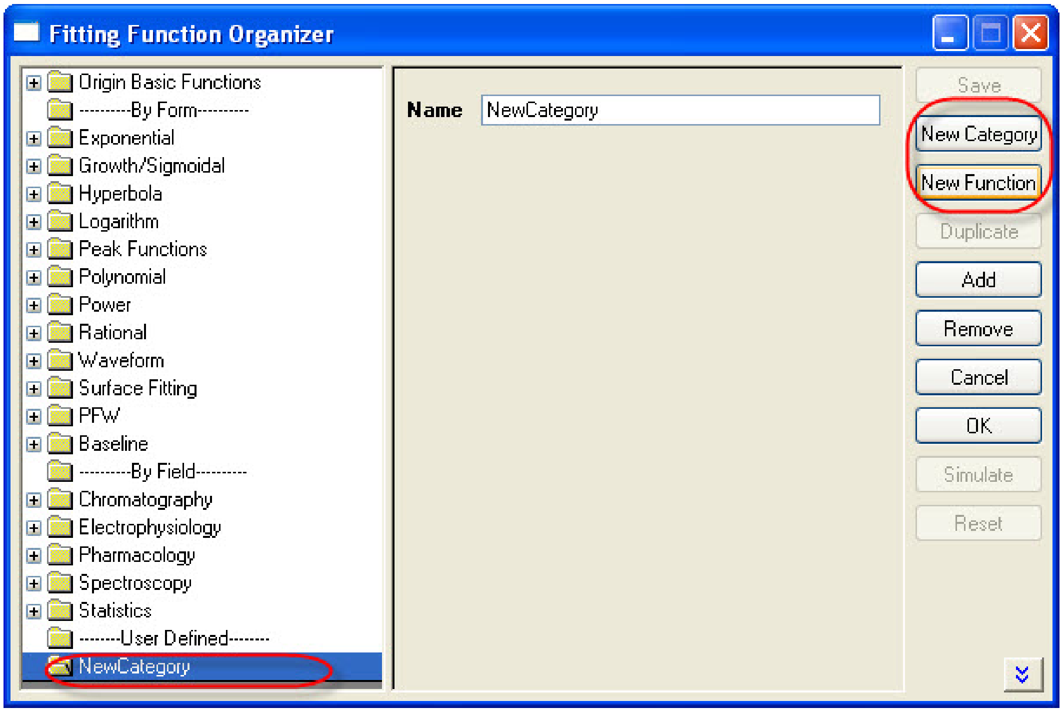
7.4.3 Example

We will illustrate how to define the following fitting function:

$$y = y0 + ae^{bx}$$

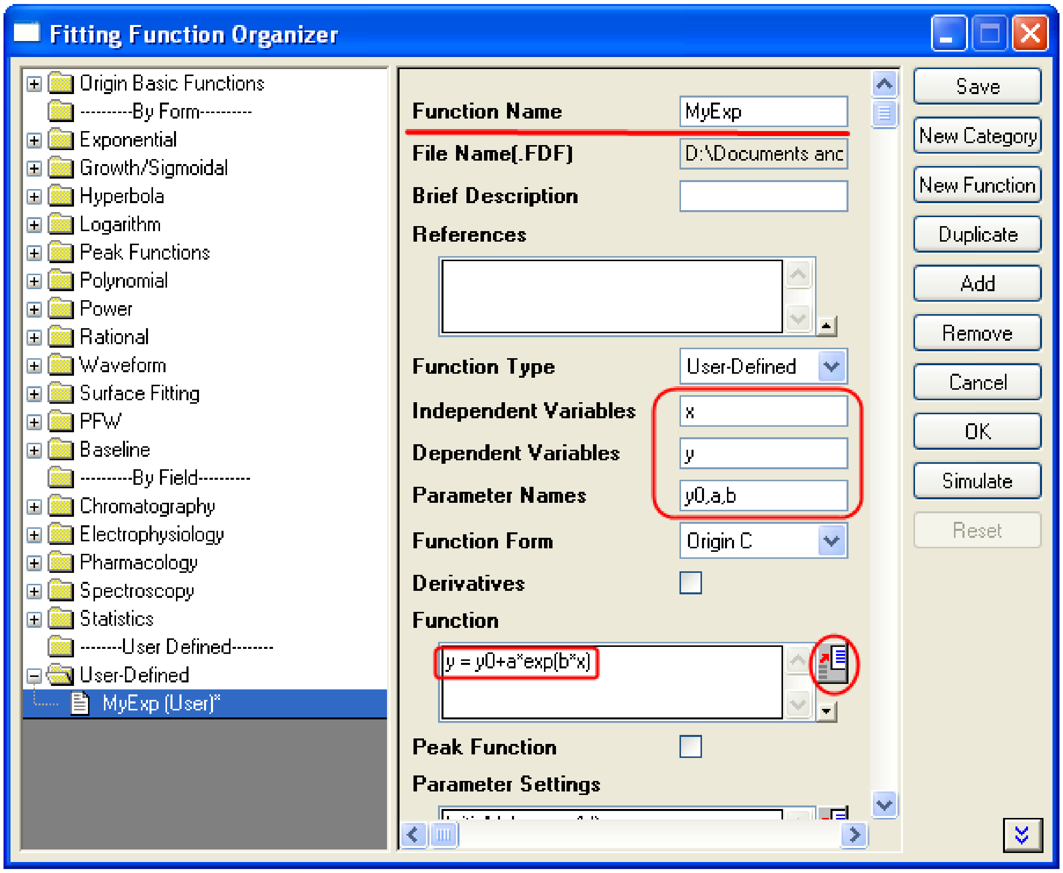
Steps to define the function:

1. Select **Tools: Fitting Function Organizer** from menu (or press **F9**) to open the function organizer. Click the **New Category** button to create a function category, rename is as *User-Defined* for example. Then press **New Function** button to create a **new function** under this category:



Name NewCategory

2. Enter function definition like the following image and **Save**:



Function Name: MyExp

File Name(.FDF): D:\Documents anc

Brief Description:

References:

Function Type: User-Defined

Independent Variables: x

Dependent Variables: y

Parameter Names: y0,a,b

Function Form: Origin C

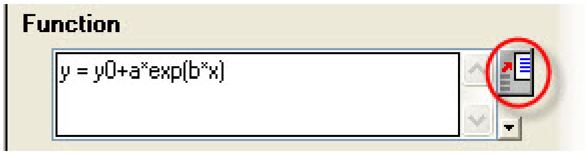
Derivatives: □

Function: y = y0+a*exp(b*x)

Peak Function: □

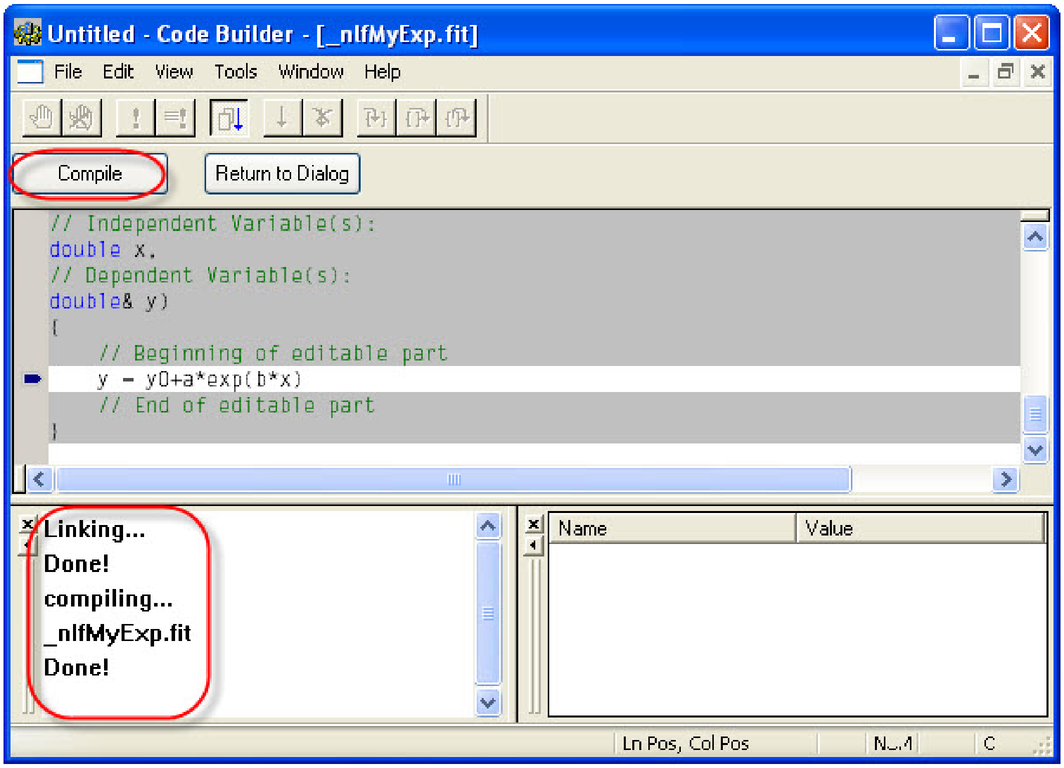
Parameter Settings

3. To verify the correctness of the function, click the button beside the **Function** box to open Origin Code Builder:



```
y = y0+a*exp(b*x)
```

In the **Code Builder**, click **Compile** button to compile the function. If passed, click **Return to Dialog** button to return **Fitting Function Organizer**.



```
// Independent Variable(s):
double x,
// Dependent Variable(s):
double& y)
{
    // Beginning of editable part
    y = y0+a*exp(b*x)
    // End of editable part
}
```

```
Linking...
Done!
compiling...
_nlfMyExp.fit
Done!
```

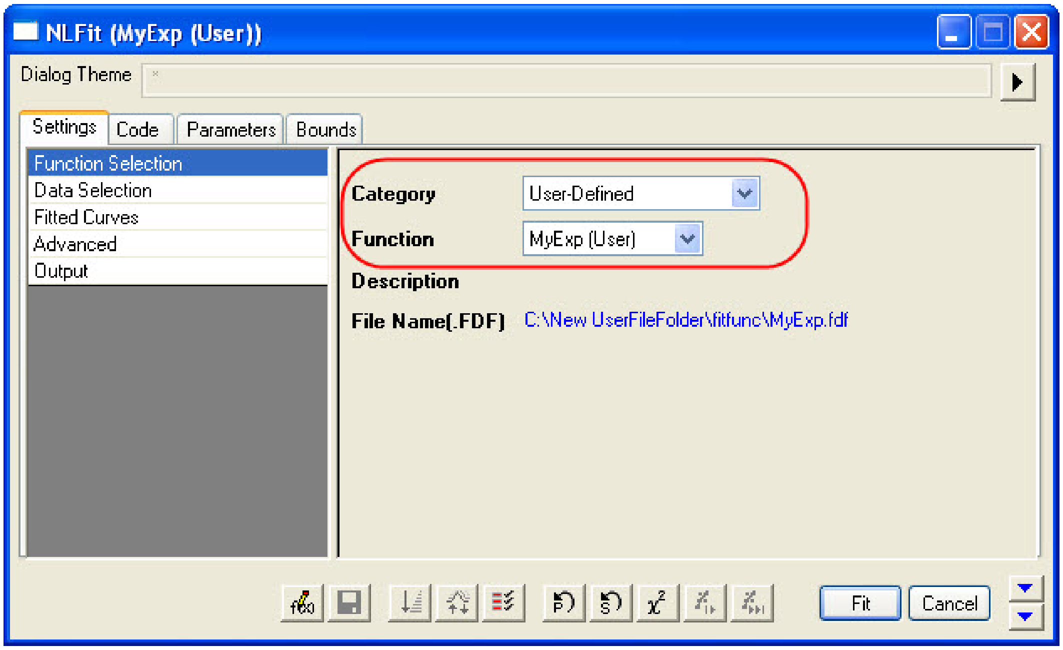
4. Click **Save** and **OK** to save the function and quite **Fitting Function Organizer**.

Fit data by the function:

1. Import \Samples\Curve Fitting\Exponential Decay.dat to Origin worksheet.

2. Highlight column B and select **Analysis: Fitting: Non-linear Curve Fit** from menu to bring up the **NLFit** dialog.

3. Select the function just defined in **Settings** tab, **Function Selection** page:



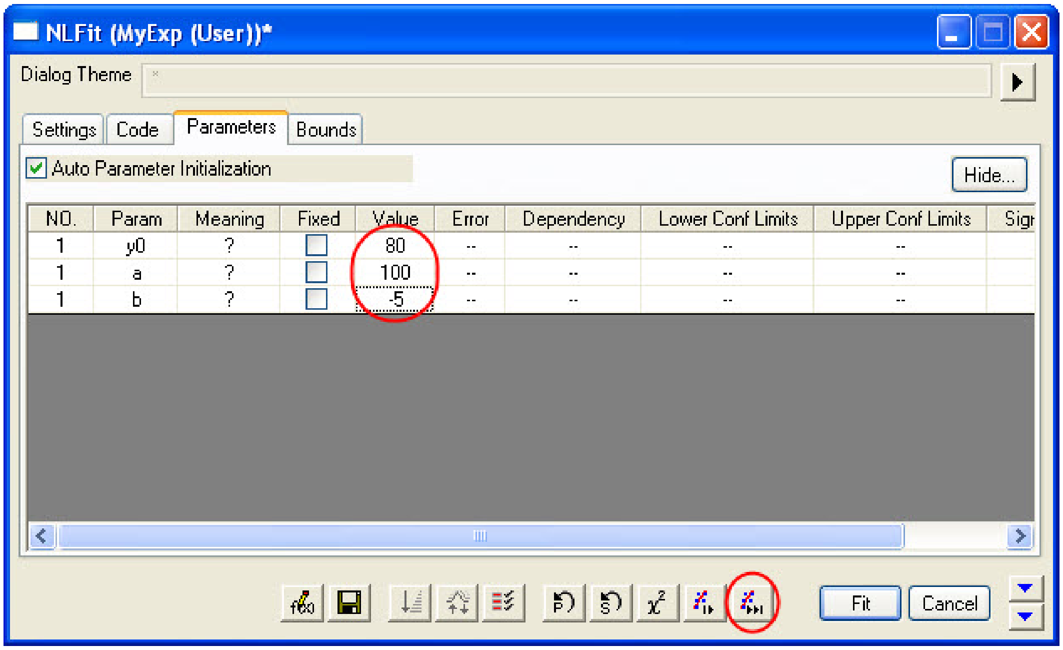
Category User-Defined

Function MyExp (User)

Description

File Name(.FDF) C:\New UserFileFolder\fitfunc\MyExp.fdf

4. Switch to **Parameters** tab, enter *80*, *100*, *-5* on the **Value** column as initial values for *y0*, *a*, *b*:

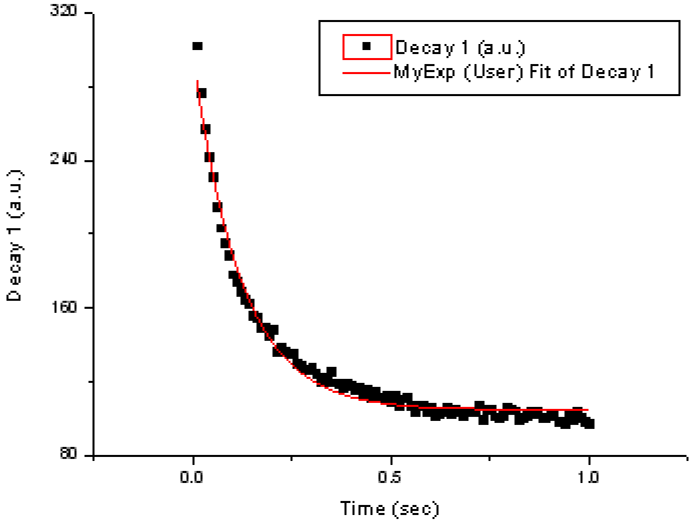


☑ Auto Parameter Initialization

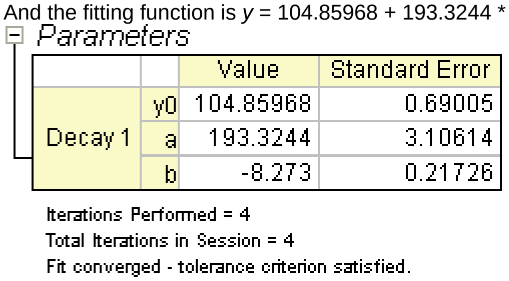
NO.	Param	Meaning	Fixed	Value	Error	Dependency	Lower Conf Limits	Upper Conf Limits	Sigr
1	y0	?	□	80	--	--	--	--	
1	a	?	□	100	--	--	--	--	
1	b	?	□	-5	--	--	--	--	

5. When the fit converged, click **OK** button to generate fitting reports.

From the Fitted Curves Plot we see the fitting is fine.



And the fitting function is *y* = 104.85968 + 193.3244 * *exp*(- 8.273 * *x*)



Parameters

		Value	Standard Error
Decay 1	y0	104.85968	0.69005
	a	193.3244	3.10614
	b	-8.273	0.21726

Iterations Performed = 4
Total Iterations in Session = 4
Fit converged - tolerance criterion satisfied.

7.5 Fitting One Dataset as a Function of Other Datasets

7.5.1 Summary

Sometimes, one may want to perform "Dataset Fitting", that is, the output may be composed of one or several datasets, like:

$$Output = A_1 * Dataset_1 + A_2 * Dataset_2$$

For example, you may want to analyze a composite spectrum to find the contributions/ratio from individual component spectra. This can be accomplished either by defining multiple independent variables or by calculating the "combination" inside the fitting function.

Minimum Origin Version Required: Origin 8.0 SR6

7.5.2 What you will learn

This tutorial will show you how to:

- Perform "Dataset Fitting"
- Define multiple independent variable fitting function

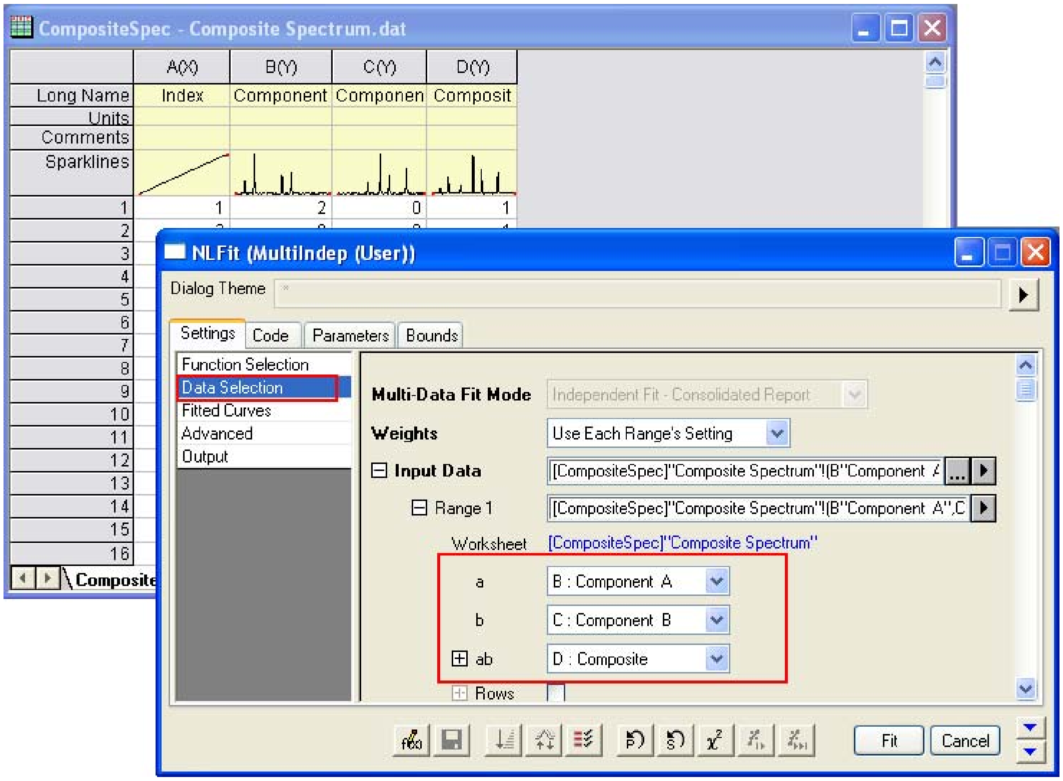
7.5.3 Steps

Import the *Composite Spectrum.dat* file from the *\Samples\Curve Fitting* folder. In this sample data, we can see that column A is the index, columns B and C are the values for the spectrum of components A and B. Column D contains values obtained after reading a composite spectrum of components A and B. By fitting column D to an equation determined by the component spectra of the pure forms of columns B and C, the coefficients for the contributions of B and C (call them c1 and c2 respectively) can be found. (Note: In this case, we supposed the independent and dependent variables have the same size. If not, interpolation is need.)

Bring up the **Fitting Function Organizer** and define a new fitting function as follow:

Function Name:	MultiIndep
Function Type:	User-Defined
Independent Variables:	a, b
Dependent Variables:	ab
Parameter Names:	C1, C2
Function Form:	Origin C
Function:	ab = C1*a + C2*b;

Initialize both C1 and C2 to 1 in the **Parameter Initialization** edit box by entering:
C1=1;
C2=1;
Save the fitting function and close **Fitting Function Organizer**. Highlight **ONLY** Column D and bring up the **NLFit** dialog, specify the input datasets in the **Data Selection** page as follow:



CompositeSpec - Composite Spectrum.dat

	A(X)	B(Y)	C(Y)	D(Y)
Long Name	Index	Component	Componen	Composit
Units				
Comments				
Sparklines				
1	1	2	0	1

NLFit (MultiIndep (User))

Dialog Theme *

Multi-Data Fit Mode: Independent Fit - Consolidated Report
Weights: Use Each Range's Setting
Input Data: [CompositeSpec]"Composite Spectrum"!(B"Component A
Range 1: [CompositeSpec]"Composite Spectrum"!(B"Component A",C
Worksheet: [CompositeSpec]"Composite Spectrum"
a: B : Component A
b: C : Component B
ab: D : Composite
Rows

Then you can click the **Fit** button to generate results.

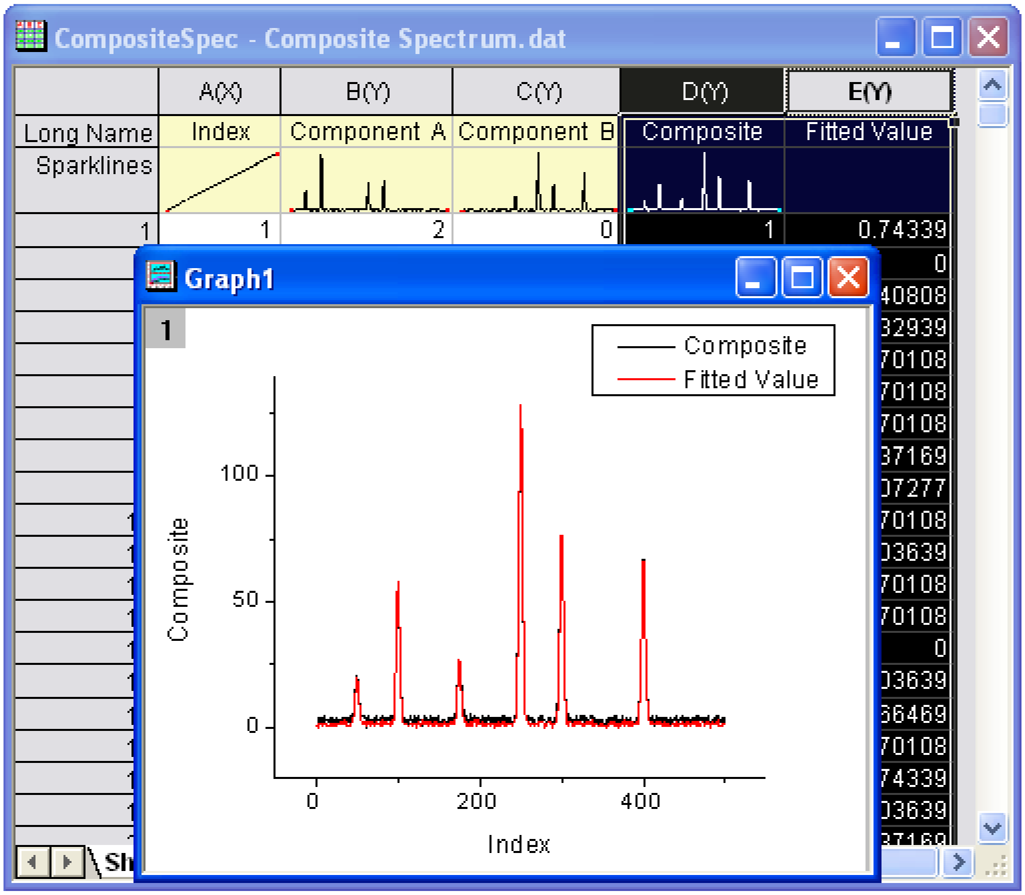
7.5.4 Results

You are supposed to get these results:

	Value	Standard Error

C1	0.37169	0.00483
C2	0.66469	0.0047

To verify the fitted results, you can add a new column and Copy + Paste the fitted value, which comes from the fitted Y in the worksheet *FitNLCurve1*, into it. Then Highlight the *Composite* and the fitted data and plot a line graph to see how good the fit is:



7.6 Fitting With Multiple Independent Variables

7.6.1 Summary

The **Function Organizer** tool can be used to create user-defined functions with more than one independent or dependent variable. The **NLFit dialog** can then be used to fit with such functions. The preview window in the fitter dialog is capable of plotting only one quantity versus another, however even if the preview does not *make sense*, the fitting process will correctly proceed once proper data and parameter assignments have been made.

Note that if you wish to fit multiple independent variables with an equation of the type
*y = A0 + A1 * x1 + A2 * x2 + ...*
you can make use of the Multiple Regression tool instead of the nonlinear fitter dialog.

Minimum Origin Version Required: Origin 8.0 SR6

7.6.2 What you will learn

This tutorial will show you how to:

- Create a user-defined fitting function with two independent variables and one dependent variable

- Fit with that function in NLFit

7.6.3 Steps

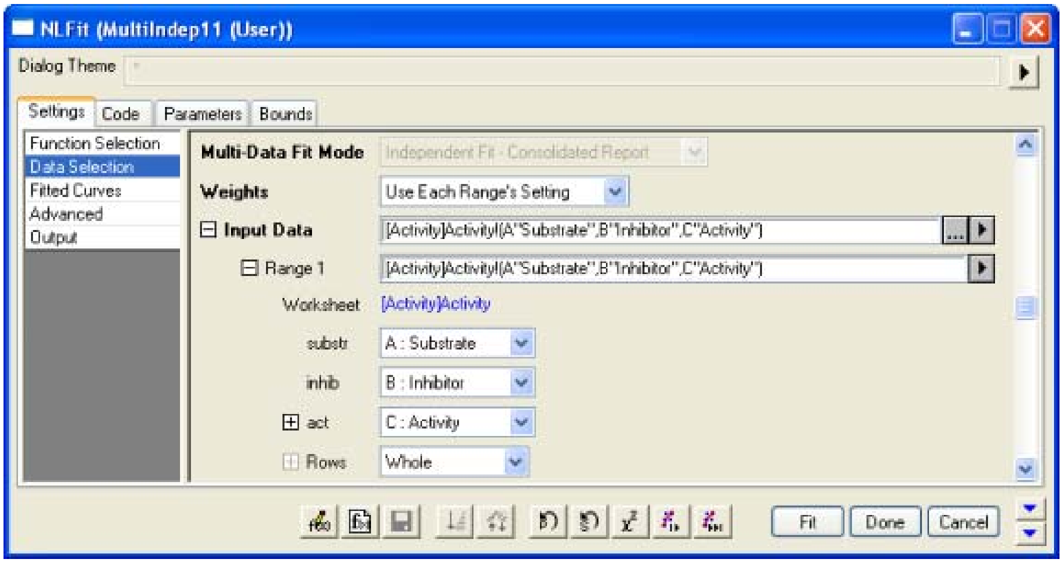
1. Start with a new workbook and import the file *\Samples\Curve Fitting\Activity.dat*.
2. Select **Tools: Fitting Function Organizer** from menu (or press **F9**) to bring up the **Fitting Function Organizer** and define a new fitting function named **MultiIndep** in **NewCategory** (create the category if not exist) as follow:

Function Name:	MultiIndep
Function Type:	User-Defined
Independent Variables:	substr,inhib
Dependent Variables:	act
Parameter Names:	ki,km,vm
Function Form:	Origin C

Function:	double mix = inhib / ki;
	act = vm * substr / (km + (1 + mix) * substr);

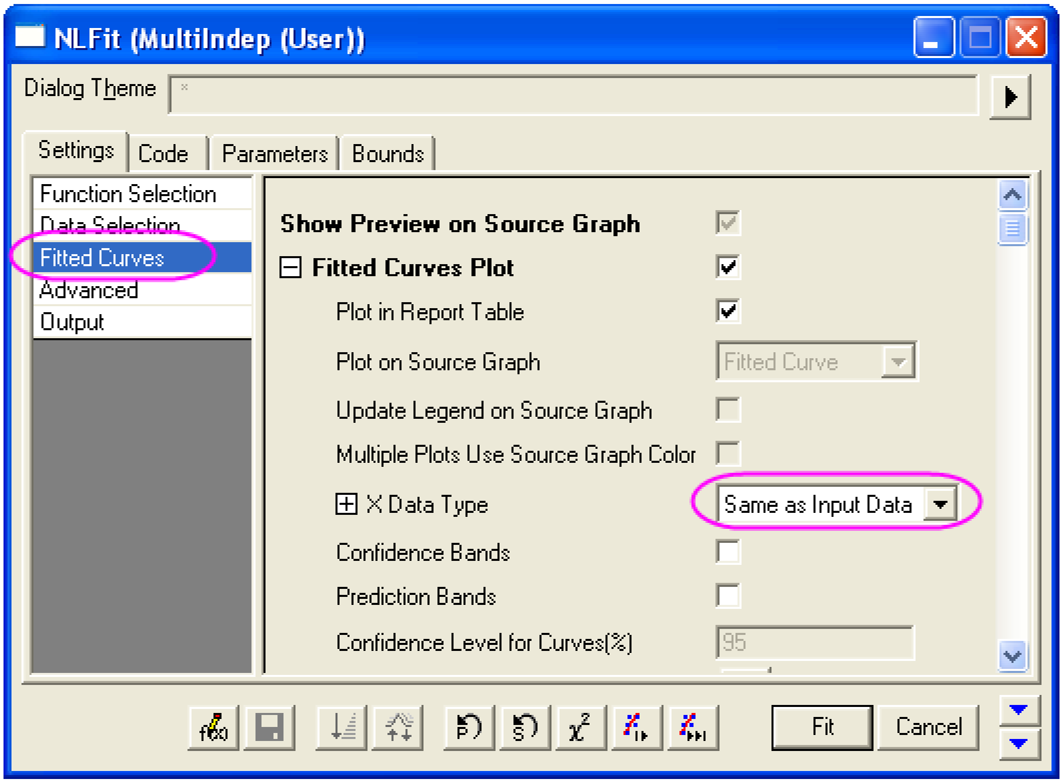
3. For more details about User Defined Fitting Function please refer to User Defined Fitting Function using Origin C.

4. Highlight **ONLY** column C and select **Analysis: Fitting: Non-linear Curve Fit** from menu to bring up the NLFit dialog. Select the function **MultiIndep** from **NewCategory** on the **Settings: Function Selection** page. Set the input datasets in the **Data Selection** page as follow:



Multi-Data Fit Mode: Independent Fit - Consolidated Report
Weights: Use Each Range's Setting
Input Data: [Activity]Activity!(A"Substrate",B"Inhibitor",C"Activity")
Range 1: [Activity]Activity!(A"Substrate",B"Inhibitor",C"Activity")
Worksheet: [Activity]Activity
substr: A : Substrate
inhib: B : Inhibitor
act: C : Activity
Rows: Whole

5. Select the **Fitted Curves** page and expand the **Fitted Curves Plot** branch. Then select **Sample as Input Data** from the drop-down list next to the **X Data Type** branch.



NLFit (MultiIndep (User))

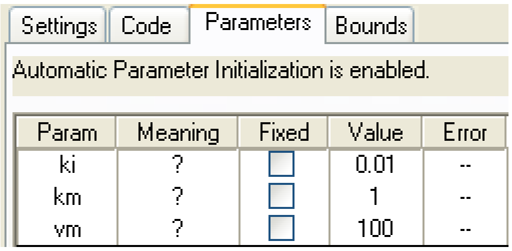
Dialog Theme *

Settings | Code | Parameters | Bounds

- Function Selection
- Data Selection
- Fitted Curves
- Advanced
- Output

Show Preview on Source Graph	☑
⊟ **Fitted Curves Plot**	☑
Plot in Report Table	☑
Plot on Source Graph	Fitted Curve
Update Legend on Source Graph	□
Multiple Plots Use Source Graph Color	□
⊞ X Data Type	Same as Input Data
Confidence Bands	□
Prediction Bands	□
Confidence Level for Curves(%)	95

6. Select **Parameters Tab** and set the initial values as follow:

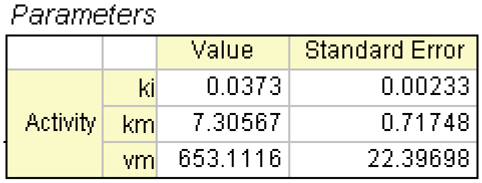


Settings | Code | Parameters | Bounds

Automatic Parameter Initialization is enabled.

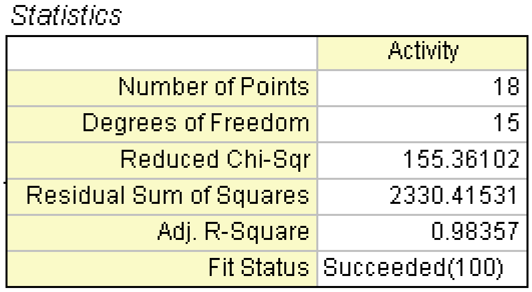
Param	Meaning	Fixed	Value	Error
ki	?	□	0.01	--
km	?	□	1	--
vm	?	□	100	--

7. Click **Fit** button to generate the fitting reports. You can see these results from the report worksheet as below:



Parameters

		Value	Standard Error
Activity	ki	0.0373	0.00233
	km	7.30567	0.71748
	vm	653.1116	22.39698



Statistics

	Activity
Number of Points	18
Degrees of Freedom	15
Reduced Chi-Sqr	155.36102
Residual Sum of Squares	2330.41531
Adj. R-Square	0.98357
Fit Status	Succeeded(100)

From the Statistics table we can know that the fitting is fairly successful.

7.7 User Defined Fitting Function using GNU Scientific Library

This article demonstrates how to use GSL function as fit function.

Minimum Origin Version Required: Origin 8.0 SR6

1. We will fit the sample Data below by the following model:

$$y = y_0 + a \int_0^x e^{\beta \cdot t}\, dt$$

0.1	0.10517
0.2	0.2214
0.3	0.34986
0.4	0.49182
0.5	0.64872
0.6	0.82212
0.7	1.01375
0.8	1.22554
0.9	1.4596
1	1.71828
1.1	2.00417
1.2	2.32012
1.3	2.6693
1.4	3.0552
1.5	3.48169
1.6	3.95303
1.7	4.47395
1.8	5.04965
1.9	5.68589
2	6.38906
2.1	7.16617
2.2	8.02501
2.3	8.97418
2.4	10.02318
2.5	11.18249
2.6	12.46374
2.7	13.87973

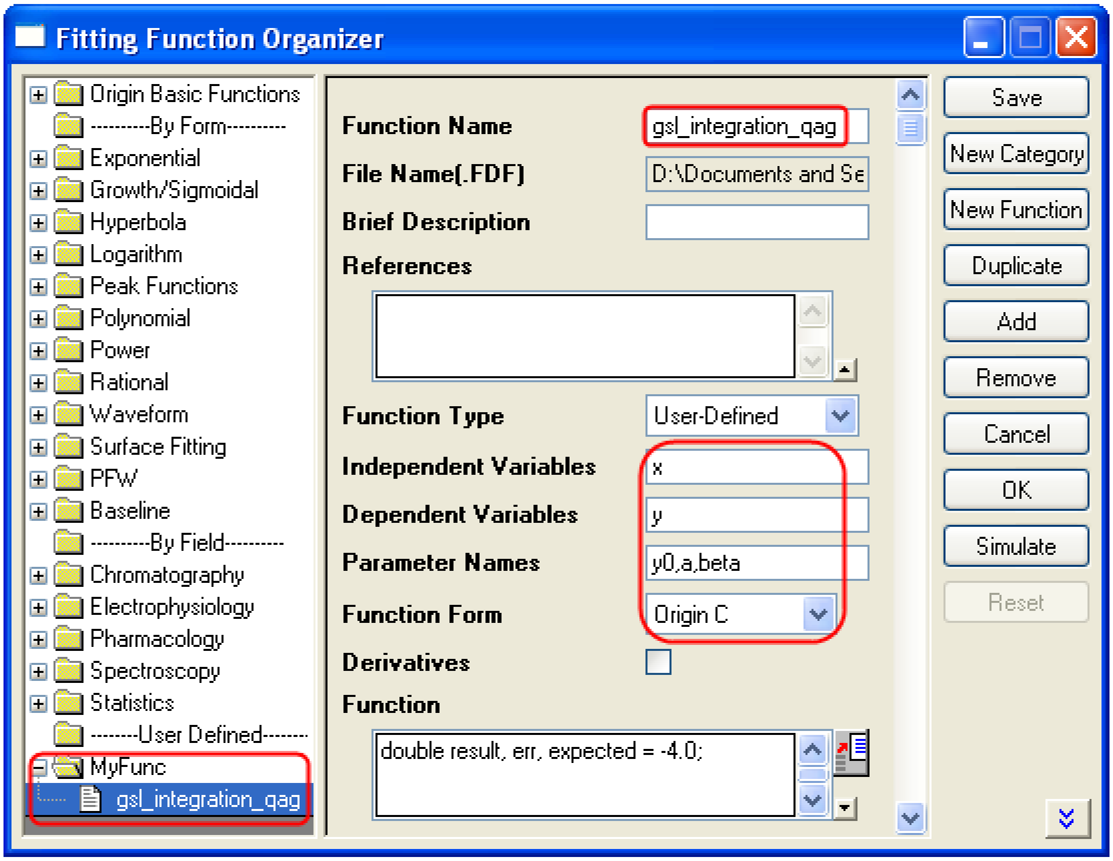
```
2.8     15.44465
2.9     17.17415
3       19.08554
3.1     21.19795
3.2     23.53253
```

2. Add the file ocgsl.h in **(Origin installation folder)\OriginC**, before next step, first make sure the gsl dlls are copied to this same location, see Calling GNU Scientific Library.

ocgsl.h

```
#pragma dll(libgsl, header)
// this is OC special pragma,
// header keyword is to indicate libgsl.dll is in same location as this file

#define GSL_EXPORT     // for OC, this is not needed, so make it empty

// you can directly search and copy gsl function prototypes here

typedef double (* FUNC)(double x, void * params);

struct gsl_function_struct
{
        FUNC function;
        void * params;
};

typedef struct gsl_function_struct gsl_function ;

typedef struct
{
    size_t limit;
    size_t size;
    size_t nrmax;
    size_t i;
    size_t maximum_level;
    double *alist;
    double *blist;
    double *rlist;
    double *elist;
    size_t *order;
    size_t *level;
}
gsl_integration_workspace;

GSL_EXPORT gsl_integration_workspace *gsl_integration_workspace_alloc (const size_t n);

GSL_EXPORT void gsl_integration_workspace_free (gsl_integration_workspace * w);

GSL_EXPORT int gsl_integration_qag (const gsl_function * f,
                                    double a, double b,
                                    double epsabs, double epsrel, size_t limit,
                                    int key,
                                    gsl_integration_workspace * workspace,
                                    double *result, double *abserr);
```

3. Press F9 to open the **Fitting Function Organizer** and then add a new function as follows:



Function Name: gsl_integration_qag

File Name(.FDF): D:\Documents and Se

Brief Description:

References:

Function Type: User-Defined

Independent Variables: x

Dependent Variables: y

Parameter Names: y0,a,beta

Function Form: Origin C

Derivatives: □

Function: double result, err, expected = -4.0;

4. Press the button on the right hand side of the **Function** Field to open the code builder and add the following codes and compile: **_nlfgsl_integration_qag.fit**

```
#include <..\ocgsl.h>

static double f_callback(double x, void * params)
{
        double alpha = *(double *)params;
        return exp(alpha*x);
}

void _nlsfgsl_integration_qag(
// Fit Parameter(s):
double y0, double a, double beta,
// Independent Variable(s):
double x,
// Dependent Variable(s):
double& y)
{
        // Beginning of editable part
        double result, err, expected = -4.0;

        // Allocates a workspace suffcient to hold 1000 double precision intervals,
        // their integration results and error estimates
        gsl_integration_workspace *ww = gsl_integration_workspace_alloc(1000);

        gsl_function F;
        F.function = f_callback;
        F.params = &beta ;

        // integral interval (0, x), within the desired absolute
        // error 0 and relative error 1e-7
        gsl_integration_qag(&F, 0, x, 0, 1e-7, 1000, 0, ww, &result, &err);

        // frees the memory associated with the workspace w
        gsl_integration_workspace_free (ww);

        y = y0 + a*result;

        // End of editable part
}
```

Furthermore, a more elaborate but efficient version of the fitting function is given as follows

```
//----------------------------------------------------------
//
#include <ONLSF.h>
#include <..\ocgsl.h>

static double f_callback(double x, void * params)
{
        double alpha = *(double *)params;
        return exp(alpha*x);
}

void _nlsfgsl_integration_qag(
// Fit Parameter(s):
double y0, double a, double beta,
// Independent Variable(s):
double x,
// Dependent Variable(s):
double& y)
{
        // Beginning of editable part

        NLFitContext *pCtxt = Project.GetNLFitContext();
        if ( pCtxt )
        {
               static vector vInteg;
               NLSFCURRINFO    stCurrInfo;
               pCtxt->GetFitCurrInfo(&stCurrInfo);
               int nCurrentIndex = stCurrInfo.nCurrDataIndex;

               BOOL bIsNewParamValues = pCtxt->IsNewParamValues();
               if ( bIsNewParamValues )
               {
                       vector vx;
                       pCtxt->GetIndepData(&vx);
                       int nSize = vx.GetSize();
                       vInteg.SetSize(nSize);

                       // Allocates a workspace suffcient to hold 1000 double precision
intervals,
                       // their integration results and error estimates
                       gsl_integration_workspace *ww =
gsl_integration_workspace_alloc(1000);

                       gsl_function F;
                       F.function = f_callback;
                       F.params = &beta ;

                       double result, err, expected = -4.0;
                       for(int ii=0; ii<nSize; ++ii)
                       {
                              // integral interval (0, vx[ii]), within the desired
absolute
                              // error 0 and relative error 1e-7
                              gsl_integration_qag(&F, 0, vx[ii], 0, 1e-7, 1000, 0, ww,
&result, &err);
                              vInteg[ii] = result;
                       }

                       // frees the memory associated with the workspace w
                       gsl_integration_workspace_free (ww);

               }
```

```
                y = y0 + a*vInteg[nCurrentIndex];
                x;
        }

        // End of editable part
}
```

5. Add the following initilization codes:

Parameter Init

```
//Code to be executed to initialize parameters
sort( x_y_curve );
double coeff[2];
fitpoly( x_y_curve, 1, coeff);
a = coeff[0];
y0 = coeff[1];
beta=1.0
```

6. Fit using the user-defined function **gsl_integration_qag**, here are the results:

y0 = -1.06363E-6

a = 1

beta =1

7.8 Fitting with NAG Special Function

7.8.1 Summary

Origin allows user to define an Origin C fitting function using NAG special functions. You can call NAG routine to evaluate the special function.

Minimum Origin Version Required: Origin 8.0 SR6

7.8.2 What you will learn

This tutorial will show you how to:

- Create fitting function using Fitting Function Organizer
- Create fitting function using NAG special function

7.8.3 Example and Steps

We will fit the following model:

$$inorm = A*exp(-td/2.0/(t-t0))*(I0(td/2.0/(t-t0))+I1(td/2.0/(t-t0)))$$

Here *A*, *td* and *t*0 are the model parameters we want to obtain from the data fitting. *I*0 and *I*1 are the first kind of Modified Bessel function of order 0 and order 1, respectively. For current example, we use the sample data in the end of this tutorial. The fitting procedure can be outlined into the following steps:

Press **F9** to open the **Fitting Function Organizer** and then create a new Category named ***FittingWithNAGSpecialFunc***. Define a new fitting function ***FittingWithBessel*** in the new category as follow:

Function Name:	FittingWithBessel
Function Type:	User-Defined
Independent Variables:	t

Dependent Variables: inorm

Parameter Names: A,t0,td

Function Form: Origin C

Function:

Click the button (icon) beside the **Function** box to open the code builder and define and compile and save the fitting function as follows:

```
#include <origin.h>

// Add your special include files here.
// For example, if you want to fit with functions from the NAG library,
// add the header file for the NAG functions here.
#include <OC_nag8.h>

// Add code here for other Origin C functions that you want to define in this file,
// and access in your fitting function.

// You can access C functions defined in other files, if those files are loaded and
compiled
// in your workspace, and the functions have been prototyped in a header file that you
have
// included above.

// You can access NLSF object methods and properties directly in your function code.

// You should follow C-language syntax in defining your function.
// For instance, if your parameter name is P1, you cannot use p1 in your function code.
// When using fractions, remember that integer division such as 1/2 is equal to 0, and
not 0.5
// Use 0.5 or 1/2.0 to get the correct value.

// For more information and examples, please refer to the "User-Defined Fitting Function"
// section of the Origin Help file.


//----------------------------------------------------------
//
void _nlsfFittingWithBessel(
// Fit Parameter(s):
double A, double t0, double td,
// Independent Variable(s):
double t,
// Dependent Variable(s):
double& inorm)
{
        // Beginning of editable part
        //inorm=  A* exp(-td/2.0/(t-t0)) *   ( s18aec(td/2.0/(t-
t0),NAGERR_DEFAULT)+s18afc(td/2.0/(t-t0),NAGERR_DEFAULT)             );

        static NagError fail1;
        static NagError fail2;
        double dtemp = td/2.0/(t-t0);
        inorm=  A* exp(-dtemp) * ( s18aec(dtemp,&fail1)+s18afc(dtemp,&fail2) );
        if(fail1.code !=NE_NOERROR)
                printf("%s\n",fail1.message);
        if(fail2.code !=NE_NOERROR)
                printf("%s\n",fail2.message);


        // End of editable part
}
```

Simulate the Function

After the function body is defined, you can click the **Compile** button in **Code Builder** to check syntax errors. And then click **Return to Dialog** button to go back **Fitting Function Organizer** dialog box. Now click the **Save** button to generate the .FDF file (Function definition file).

Once you have a .FDF file, you can click the **Simulate** button to simulate a curve, this will be very helpful to evaluate the initial values. In the **simcurve** dialog, enter some proper parameter values and X range, and see what the curve looks like in the **Preview** panel.

Set the Initial Values for the Parameters

As it is a user-defined fitting function, you have to supply the initial guess values for the parameters before performing your fitting task for the data. You may do it by set them manually in the **Parameter** tab in **Nonlinear Curve Fit** dialog. For the sample data shown below, you can just set the initial values for the parameters A = 1, td = 1, t0 = 1. After the parameters are initialized, you can then do the fitting to obtain the fitting result, as shown to the right of the sample data.

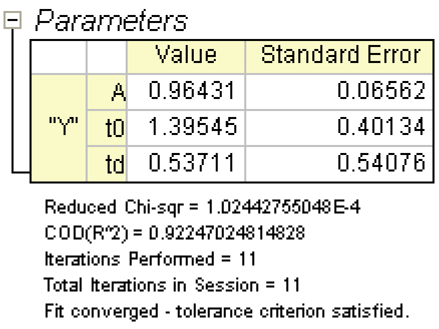
7.8.4 Sample Data

Copy the below sample data and use **Import Wizard** to import the data from **Clipboard**, then do the fitting using the given initial values for the parameters: A = 1, td = 1, t0 = 1.

Sample Data

X	Y
2	0.7868954118
2.080808081	0.8133022141
2.161616162	0.8178216765
2.242424242	0.8427866729
2.323232323	0.8315815363
2.404040404	0.8484657180
2.565656566	0.8618233553
2.646464646	0.8745962570
2.727272727	0.8921620316
2.808080808	0.8687399759

Results



Parameters

		Value	Standard Error
"Y"	A	0.96431	0.06562
	t0	1.39545	0.40134
	td	0.53711	0.54076

```
Reduced Chi-sqr = 1.02442755048E-4
COD(R^2) = 0.92247024814828
Iterations Performed = 11
Total Iterations in Session = 11
Fit converged - tolerance criterion satisfied.
```

7.9 Fitting with Integral using NAG Library

7.9.1 Summary

Origin allows user to define an Origin C fitting function which involves an integral. You can call NAG functions to perform the integration while defining the fitting function. There are built-in functions in Origin C which perform integration. For the current example, the NAG solution is recommended. It has a better performance compared to the built-in integration algorithm. Note that an **infinite NAG integrator** is used here.

Minimum Origin Version Required: Origin 8.0 SR6

7.9.2 What you will learn

This tutorial will show you how to:

- Create a fitting function using the Fitting Function Organizer
- Create a fitting function with a Definite Integral using a NAG integration routine
- Set up the Initial Code for the fitting function

7.9.3 Example and Steps

We will fit the following model:

$$y = y_0 + \int_{-\infty}^{x} \frac{A}{w\sqrt{\frac{\pi}{2}}} e^{-2\frac{(x-x_c)^2}{w^2}}, dx$$

Here y_0 *A*, *xc* and *w* are the model parameters we want to obtain from the data fitting. The fitting procedure can be outlined into the following steps:

Define the Function

Press **F9** to open the **Fitting Function Organizer** and then create a new Category named ***FittingWithIntegral***. Define a new fitting function ***nag_integration_fitting*** in the new category as follow:

Function Name:	nag_integration_fitting
Function Type:	User-Defined
Independent Variables:	x
Dependent Variables:	y
Parameter Names:	y0, A, xc, w
Function Form:	Origin C
Function:	

Click the button (icon) beside the **Function** box to open the code builder and define and compile and save the fitting function as follows:

```
#include <origin.h>
// Add your special include files here.
// For example, if you want to fit with functions from the NAG library,
// add the header file for the NAG functions here.
#include <oc_nag8.h>


// Add code here for other Origin C functions that you want to define in this file,
// and access in your fitting function.

struct user   // parameters in the integrand
{
        double amp, center, width;

};
// Function supplied by user, return the value of the integrand at a given x.
static double NAG_CALL f_callback(double x, Nag_User *comm)
{
        struct user *sp = (struct user *)(comm->p);

        double amp, center, width;    // temp variable to accept the parameters in the
Nag_User communication struct
        amp = sp->amp;
        center = sp->center;
        width = sp->width;

        return amp * exp( -2*(x - center)*(x - center)/width/width ) / (width*sqrt(PI/2));
}


// You can access C functions defined in other files, if those files are loaded and
compiled
// in your workspace, and the functions have been prototyped in a header file that you
have
// included above.

// You can access NLSF object methods and properties directly in your function code.

// You should follow C-language syntax in defining your function.
// For instance, if your parameter name is P1, you cannot use p1 in your function code.
// When using fractions, remember that integer division such as 1/2 is equal to 0, and
not 0.5
// Use 0.5 or 1/2.0 to get the correct value.

// For more information and examples, please refer to the "User-Defined Fitting Function"
// section of the Origin Help file.


//----------------------------------------------------------
//
void _nlsfnag_integration_fitting(
// Fit Parameter(s):
double y0, double A, double xc, double w,
// Independent Variable(s):
double x,
// Dependent Variable(s):
double& y)
{
        // Beginning of editable part

        // Through the absolute accuracy epsabs, relative accuracy epsrel and
max_num_subint you can
        // control the precision of the integration you need
        // if epsrel is set negative, the absolute accuracy will be used.
        // Similarly, you can control only relative accuracy by set the epsabs negative
        double epsabs = 0.0, epsrel = 0.0001;

        // The max number of sub-intervals needed to evaluate the function in the integral
        // The more diffcult the integrand the larger max_num_subint should be
        // For most problems 200 to 500 is adequate and recommmended
        Integer max_num_subint = 200;

        // Result keeps the approximate integral value returned by the algorithm
        // abserr is an estimate of the error which should be an upper bound for the |I -
result|
        // where I is the integral value
        double result, abserr;

        // The structure of type Nag_QuadProgress,
        // it contains pointers allocated memory internally with max_num_subint elements
        Nag_QuadProgress qp;

        // The NAG error parameter (structure)
        static NagError fail;

        // Parameters passed to integrand by Nag_User communication struct
         Nag_User comm;
        struct user s;
         s.amp = A;
         s.center = xc;
         s.width = w;
         comm.p = (Pointer)&s;

        // Perform integration
        // There are 3 kinds of infinite boundary types you can use in Nag infinite
integrator
        // Nag_LowerSemiInfinite, Nag_UpperSemiInfinite, Nag_Infinite
        d01smc(f_callback, Nag_LowerSemiInfinite, x, epsabs, epsrel, max_num_subint,
&result, &abserr, &qp, &comm, &fail);

         // you may want to exam the error by printing out error message, just uncomment
the following lines
        // if (fail.code != NE_NOERROR)
         // printf("%s\n", fail.message);

        // For the error other than the following three errors which are due to bad input
parameters
        // or allocation failure  NE_INT_ARG_LT  NE_BAD_PARAM   NE_ALLOC_FAIL
        // You will need to free the memory allocation before calling the integration
routine again to avoid memory leakage
```

```
        if (fail.code != NE_INT_ARG_LT && fail.code != NE_BAD_PARAM && fail.code !=
NE_ALLOC_FAIL)
        {
                NAG_FREE(qp.sub_int_beg_pts);
                NAG_FREE(qp.sub_int_end_pts);
                NAG_FREE(qp.sub_int_result);
                NAG_FREE(qp.sub_int_error);
        }

        // Calculate the fitted value
        y = y0 + result;

        // End of editable part
}
```

In the above code, we firstly define the integrand as a callback function ***f_callback*** just outside the fitting function body ***_nlsfnag_integration_fitting***. Note that we parametrize the integrand function with the variables ***amp***, ***center*** and ***width***, and pass them into the callback funtion through the *Nag_User* struct. Inside the fitting function, we perform the integration using NAG integrator ***d01smc***.

Calling NAG functions should be more efficient than writing your own routines. Using an analogous method, you can perform finite, infinite, one-dimension and multi-dimension quadrature in your fitting function. Please read the NAG Quadrature page and select a proper routine.

Simulate the Function

After entering the function body codes, you can click the **Compile** button in **Code Builder** to check syntax errors. And then click **Return to Dialog** button to go back **Fitting Function Organizer** dialog box. Now click the **Save** button to generate the .FDF file (Function definition file).

Once you have a .FDF file, you can click the **Simulate** button to simulate a curve, this will be very helpful to evaluate the initial values. In the **simcurve** dialog, enter some proper parameter values and X range, and see what the curve looks like in the **Preview** panel.

Fit the Curve

Before you start to fit the curve, it is very helpful to simulate the function first. Performing integration may take some time, if there is any mistake, you may see Origin "freeze" after you click the **Fit** button. So in the **Fitting Function Organizer** dialog, select the function we defined and click the **Simulate** button. This will bring up the **simcurve** X-Function. Enter some "guess" values and click the **Apply** button. If the simulated curve looks like your source data, you can go further to fit.

To test the fitting function, import *\Samples\Curve Fitting\Replicate Response Data.dat* to Origin. Set *Col*(*A*) = *log*(*Col*(*A*)) in the **Set Column Values** dialog. This will make a sigmoid curve. Highlight column A and B and create a scatter plot. Then bring up the **NLFit** dialog from **Analysis: Fitting: Nonlinear Curve Fit** menu item. Select the fitting function we just defined and go to the **Parameters** tab, initialize all parameters by 1 and fit. You are supposed to see these results:

	Value	Standard Error
y0	-0.00806	0.18319
A	3.16479	0.39624
xc	-0.19393	0.10108
w	1.77252	0.33878

7.10 Fitting Integral Function with parametric limit using NAG Library

7.10.1 Summary

Before you start delving into this tutorial, you are recommended to read the relevant tutorial in Fitting with Integral using NAG Library. And as far as programming is concerned, the two tutorials are basically the same, except that here you will learn to define Origin C fitting function with fitting parameters in the integral limit, while in the previous tutorial we in fact define a fitting independent variable in the integral limit. Also note that **a different NAG integrator** is used here.

Minimum Origin Version Required: Origin 8.0 SR6

7.10.2 What you will learn

This tutorial will show you how to:

- Create a fitting function with Definite Integral using the NAG integration routine
- Create a fitting function with a parametric integral limit
- Use a log function to scale a large return value from the fitting function

7.10.3 Example and Steps

For example, we will fit the sample data at the bottom of this page with the following model:

$$y = \int_c^d \frac{\cosh\left((x_i + b^2 \cdot x^2)/(b + x)\right)}{a + (x_i^2 + x^2)}\, dx_i$$

Note that we use x_i to indicate the integral independent variable while x indicates the fitting independent variable. The model parameters *a*, *b*, *c*, and *d* are fitted parameters we want to obtain from the sample data. To prepare the data, you just need to copy the sample data to an Origin Work sheet. The fitting procedure is similar to the previous tutorial:

Define Fitting Function in Fitting Function Organizer

Press **F9** to open the Fitting Function Organizer and add the User-Defined integral fitting function ***nag_integration_fitting_cosh*** to the Category ***FittingWithIntegral***, similar to the first tutorial.

Function Name:	nag_integration_fitting_cosh
Function Type:	User-Defined
Independent Variables:	x
Dependent Variables:	y
Parameter Names:	a, b, c, d
Function Form:	Origin C
Function:	

Click the button (icon) beside the **Function** box to open the code builder and define and compile the fitting function as follows: (Note: Remember to save the Function after compiling it and returning to the Function Organizer Dialog):

```
#include <origin.h>
// Add your special include files here.
// For example, if you want to fit with functions from the NAG library,
// add the header file for the NAG functions here.
#include <oc_nag8.h>


// Add code here for other Origin C functions that you want to define in this file,
// and access in your fitting function.
struct user
```

```
{
        double a, b, fitX;  // fitX the independent variable of fitting function

};
static double NAG_CALL f_callback(double x, Nag_User *comm)  // x is the independent
variable of the integrand
{

        struct user *sp = (struct user *)(comm->p);

        double aa, bb, fitX; // temp variable to accept the parameters in the Nag_User
communication struct
        aa = sp->a;
        bb = sp->b;
        fitX = sp->fitX;

        return cosh((x*x+bb*bb*fitX*fitX)/(bb+fitX))/(aa+(x*x+fitX*fitX));
}

// You can access C functions defined in other files, if those files are loaded and
compiled
// in your workspace, and the functions have been prototyped in a header file that you
have
// included above.

// You can access NLSF object methods and properties directly in your function code.

// You should follow C-language syntax in defining your function.
// For instance, if your parameter name is P1, you cannot use p1 in your function code.
// When using fractions, remember that integer division such as 1/2 is equal to 0, and
not 0.5
// Use 0.5 or 1/2.0 to get the correct value.

// For more information and examples, please refer to the "User-Defined Fitting Function"
// section of the Origin Help file.

//----------------------------------------------------------
//
void _nlsfnag_integration_fitting_cosh(
// Fit Parameter(s):
double a, double b, double c, double d,
// Independent Variable(s):
double x,
// Dependent Variable(s):
double& y)
{
        // Beginning of editable part
        double epsabs = 0.00001, epsrel = 0.0000001, result, abserr;
        Integer max_num_subint = 500;
        // you may use epsabs and epsrel and this quantity to enhance your desired
precision
        // when not enough precision encountered

        Nag_QuadProgress qp;
        static NagError fail;

        // the parameters parameterize the integrand can be input to the call_back
function
        // through the Nag_User communication struct
        Nag_User comm;
        struct user s;
        s.a = a;
        s.b = b;
        s.fitX = x;
        comm.p = (Pointer)&s;

        d01sjc(f_callback, c, d, epsabs, epsrel, max_num_subint, &result, &abserr, &qp,
&comm, &fail);
```

```
         // you may want to exam the error by printing out error message, just uncomment
the following lines
        // if (fail.code != NE_NOERROR)
         // printf("%s\n", fail.message);


        // For the error other than the following three errors which are due to bad input
parameters
        // or allocation failure  NE_INT_ARG_LT  NE_BAD_PARAM   NE_ALLOC_FAIL
        // You will need to free the memory allocation before calling the integration
routine again to
         // avoid memory leakage
        if (fail.code != NE_INT_ARG_LT && fail.code != NE_BAD_PARAM && fail.code !=
NE_ALLOC_FAIL)
        {
                NAG_FREE(qp.sub_int_beg_pts);
                NAG_FREE(qp.sub_int_end_pts);
                NAG_FREE(qp.sub_int_result);
                NAG_FREE(qp.sub_int_error);
        }


        y = log(result);
         // note use log of the integral result as return as the integral result is large,
         // you are not necessary to do so

        // End of editable part
}
```

In the above code, we define the integrand as a callback function ***f_callback*** just outside the fitting function body ***_nlsfnag_integration_fitting_cosh***. Note that we parametrize the integrand function with the variables ***a***, ***b*** and ***fitX***, and pass them into the callback funtion through the *Nag_User* struct. After that we perform the integration using NAG integrator ***d01sjc***. Besides, you can also use other Quadrature Routines as you want. In the current example, we also use a log scale for the fitting function. (The sample data are already scaled by a log function)

Compile the code, return to the dialog and then Save the fitting function in the function Organizer and open the **Nonlinear Curve Fit** dialog in the **Analysis**-**Fitting** menu. You can then select this user-defined fitting function in the **Function Selection** page under **Setting** Tab.

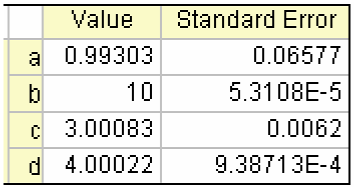
Set the Initial Values for the Parameters

Similarly, as it is a user-defined fitting function, you have to supply the initial guess values for the parameters. You may manually set them in the **Parameter** tab in **Nonlinear Curve Fit** dialog. For current example, you can just set the initial values for the parameters *a* = 1, *b* = 10, *c* = 3, *d* = 4. After the parameters are initialized, you can perform the fitting to obtain the fitting result, as shown in the following.

7.10.4 Sample Data

X	Y
-5	498.19046
-4.33333	329.43196
-3.66667	210.28005
-3	126.55799
-2.33333	69.01544
-1.66667	31.3555
-1	9.1393
-0.33333	-0.84496
0.33333	-0.99914
1	6.86736

Results:



	Value	Standard Error
a	0.99303	0.06577
b	10	5.3108E-5
c	3.00083	0.0062
d	4.00022	9.38713E-4

7.11 Fitting with Summation

7.11.1 Summary

We have showed you how to perform fitting with an integral using the NAG Library, and now you'll learn how to do that without calling NAG functions. In this tutorial, we will show you how to do integration by the trapezoidal rule and include the summation procedure in the fitting function.

Minimum Origin Version Required: Origin 8.0 SR6

7.11.2 What you will learn

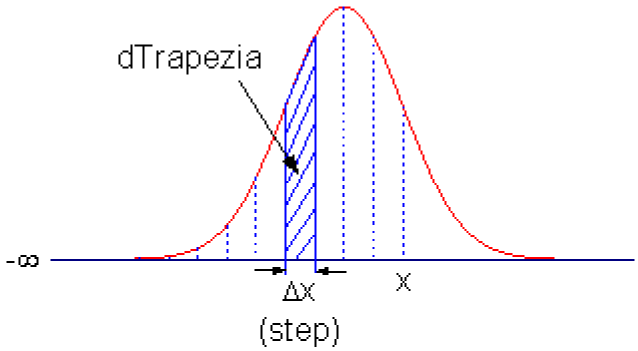
- How to include summation in your fitting function.
- Trapezoidal rule for integration.

7.11.3 Example and Steps

We will fit the same model as the integral fit using NAG:

$$y = y_0 + \int_{-\infty}^{x} \frac{A}{w\sqrt{\frac{\pi}{2}}} e^{-2\frac{(x-x_c)^2}{w^2}}\,,dx$$

The difference is that we will perform the integration within the fitting function. Using the trapezoidal rule, we will first divide the curve into pieces and then approximate the integral area by multiple trapezoids. The precision of the result then depends on how many trapezoids will be used. Since this is a semi-infinite integration, we will set an increment (steps) and construct trapezoids from the upper integral limit, x, to the lower integral limit, negative infinity and then accumulate the area of these trapezoids. When the increment of the area is significantly small, we will stop the summation. Before doing the summation, you should guarantee that the function is **CONVERGENT**, or you should include a convergence check in your code.



Define the Function

Select **Tools:Fitting Function Organizer** or alternatively press the **F9** key to open the **Fitting Function Organizer** and then define the function as follows:

Function Name:	summation
Function Type:	User-Defined
Independent Variables:	x
Dependent Variables:	y

Parameter Names: y0, A, xc, w

Function Form: Origin C

Function:

Click the button (icon) beside the **Function** box to open Code Builder. Define, compile and save the fitting function as follows:

```
#pragma warning(error : 15618)
#include <origin.h>

// Subroutine for integrand
double f(double x, double A, double xc, double w)
{
        return A * exp(-2*(x-xc)*(x-xc)/w/w) / w / sqrt(PI/2);
}

//----------------------------------------------------------
//
void _nlsfsummation(
// Fit Parameter(s):
double y0, double A, double xc, double w,
// Independent Variable(s):
double x,
// Dependent Variable(s):
double& y)
{
        // Beginning of editable part
        // Set the tolerance for stop integration.
        double dPrecision = 1e-12;
        // Initialization
        double dIntegral = 0.0;
        double dTrapezia = 0.0;
        // Steps, or Precision.
        double dStep = 0.01;
        // Perform integrate by trapezoidal rule.
        // Note that you should guarantee that the function is CONVERGENT.
        do
        {
                // Trapezia area.
                dTrapezia = 0.5 * ( f(x, A, xc, w) + f((x-dStep), A, xc, w) ) * dStep;
                // Accumulate area.
                dIntegral += dTrapezia;
                x -= dStep;
        }while( (dTrapezia/dIntegral) > dPrecision );
        // Set y value.
        y = y0 + dIntegral;
        // End of editable part
}
```

Fit the Curve

We can use the same data to test the result.

1. Import *\Samples\Curve Fitting\Replicate Response Data.dat*.
2. Highlight the first column, right-click on it, and select Set Column Values from the context menu.
3. Set *Col*(*A*) = *log*(*Col*(*A*)) in the **Set Column Values** dialog. This will make a sigmoidal curve.
4. Highlight columns A and B and create a scatter plot.
5. Then bring up the **NLFit** dialog by pressing **Ctrl** + **Y**. Select the fitting function we just defined and go to the **Parameters** tab, initialize all parameters to 1 and fit. You should see these results:

	Value	Standard Error
y0	-0.00806	0.18319
A	3.16479	0.39624

xc	-0.19393	0.10108
w	1.7725	0.33878

7.12 Fitting Complex Function

7.12.1 Summary

When fitting with a complex function, we can easily separate the complex function to two functions: one corresponding to its real part and the other corresponding to its imaginary part. With these two functions, we can define the complex fitting function with two dependent variables by **Fitting Function Organizer** and can access it in **NLFit** dialog. We will illustrate how to fit with complex function below. More details about fitting with multiple dependent or independent variable please refer to Fitting with Multiple Independent Variables.

Minimum Origin Version Required: Origin 8.0 SR6

7.12.2 What you will learn

This tutorial will show you how to:

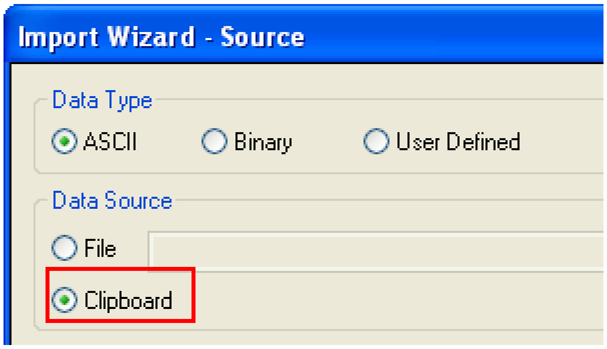
- Create a user-defined complex fitting function with two dependent variables and one independent variable
- Fit with such function in NLFit

7.12.3 Steps

1. Select whole form below (including header line) and right click to choose **Copy** to put the data in clipboard.

Omega	Y1	Y2
0	3	0
0.01	2.88462	-0.28846
0.02	2.58621	-0.51724
0.03	2.20588	-0.66176
0.04	1.82927	-0.73171
0.05	1.5	-0.75
0.06	1.22951	-0.7377
0.07	1.01351	-0.70946
0.08	0.8427	-0.67416
0.09	0.70755	-0.63679
0.1	0.6	-0.6
0.11	0.5137	-0.56507

2. Select **Import/ Import Wizard** to open Import Wizard dialog. Then choose **Clipboard** in **Data Source** group and click **Finish** to import the data.



Import Wizard - Source

Data Type

● ASCII ○ Binary ○ User Defined

Data Source

○ File

● Clipboard

3. Select **Tools: Fitting Function Organizer** from menu (or press **F9**) to bring up the **Fitting Function Organizer** and define a new fitting function named **ComplexFitting** in **NewCategory** (create the category if not exist) as follow:

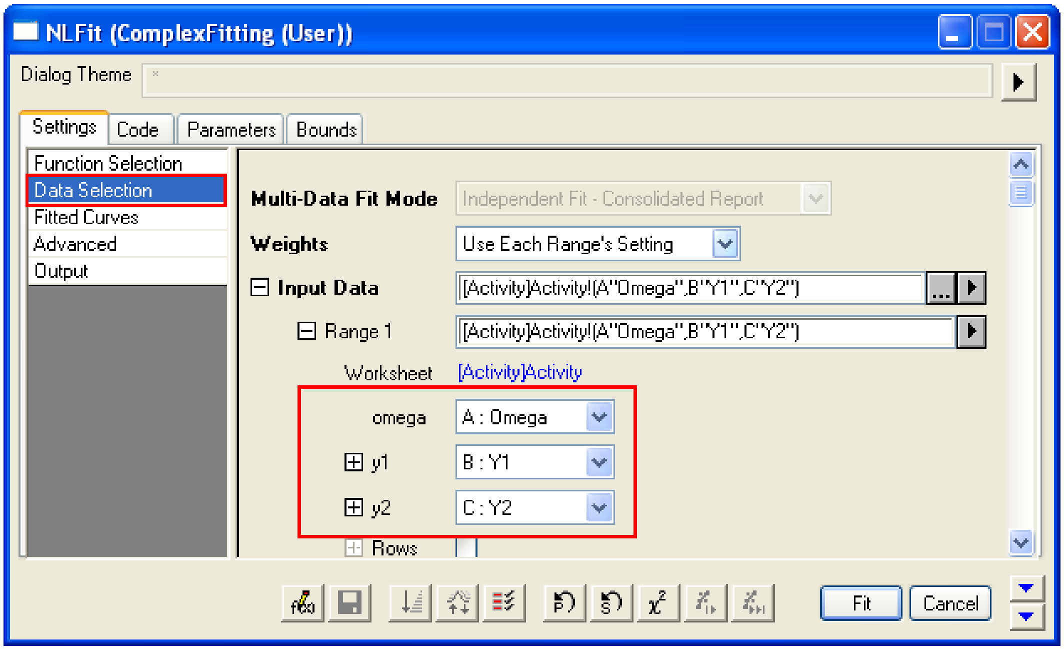
Function Name:	ComplexFitting
Function Type:	User-Defined
Independent Variables:	omega
Dependent Variables:	y1,y2
Parameter Names:	A,tau
Function Form:	Origin C

Function:	complex cc = A/(1+1i*omega*tau);
	y1 = cc.m_re;
	y2 = cc.m_im;

4. Note: To use the imaginary unit "i" for creating complex numbers, you need to write it as "1i" in Origin C, as in the above **Function** row. And **complex** is a class that implements a complex number data type. It contains both a Real and an Imaginary component.

5. For more details about creating user-defined fitting function, please refer to User Defined Fitting Function using Origin C.

6. Highlight all the columns and select **Analysis: Fitting: Non-linear Curve Fit** from menu to bring up the NLFit dialog. Select the function **ComplexFitting** from **NewCategory** on the **Settings: Function Selection** page. Set the input datasets in the **Data Selection** page as follow:



NLFit (ComplexFitting (User))

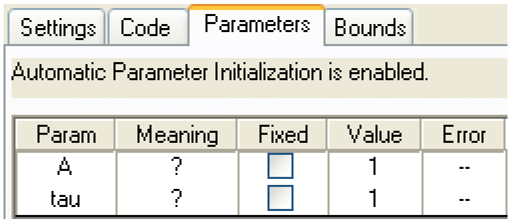
Dialog Theme *

Settings | Code | Parameters | Bounds

Function Selection
Data Selection
Fitted Curves
Advanced
Output

Multi-Data Fit Mode	Independent Fit - Consolidated Report
Weights	Use Each Range's Setting
Input Data	[Activity]Activity!(A"Omega",B"Y1",C"Y2")
Range 1	[Activity]Activity!(A"Omega",B"Y1",C"Y2")
Worksheet	[Activity]Activity
omega	A : Omega
y1	B : Y1
y2	C : Y2
Rows	

7. Select **Parameters Tab** and set the initial values as follows:

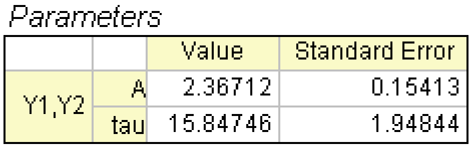


Settings | Code | Parameters | Bounds

Automatic Parameter Initialization is enabled.

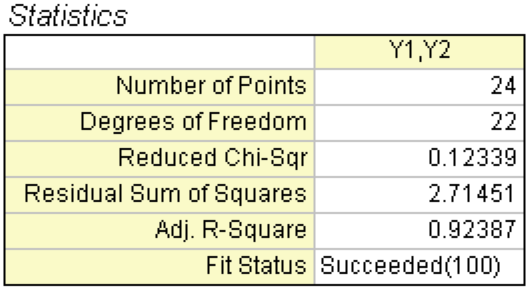
Param	Meaning	Fixed	Value	Error
A	?	□	1	--
tau	?	□	1	--

8. Click **Fit** to generate the fitting report sheet. You can see the results from the report worksheet as below:



Parameters

		Value	Standard Error
Y1,Y2	A	2.36712	0.15413
	tau	15.84746	1.94844



Statistics

	Y1,Y2
Number of Points	24
Degrees of Freedom	22
Reduced Chi-Sqr	0.12339
Residual Sum of Squares	2.71451
Adj. R-Square	0.92387
Fit Status	Succeeded(100)

From the Statistics table, we can see that the fitting is fairly successful.

7.13 Fitting with Convolution

7.13.1 Summary

When performing curve fitting to experimental data, one may encounter the need to account for instrument response in the data. One way to do this is to first perform deconvolution on the data to remove the instrument response, and then perform curve fitting as a second step. However deconvolution is not always reliable as the results can be very sensitive to any noise present in the data. A more reliable way is to perform convolution of the fitting function with the instrument response while performing the fitting. This tutorial will demonstrate how to perform convolution while fitting.

Minimum Origin Version Required: Origin 8 SR6.

7.13.2 What you will learn

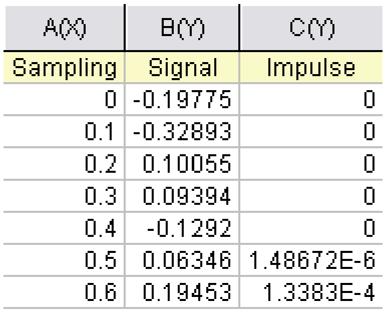
This tutorial will show you how to:

- access fitting information during iterations.
- perform convolution while fitting.

7.13.3 Example and Steps

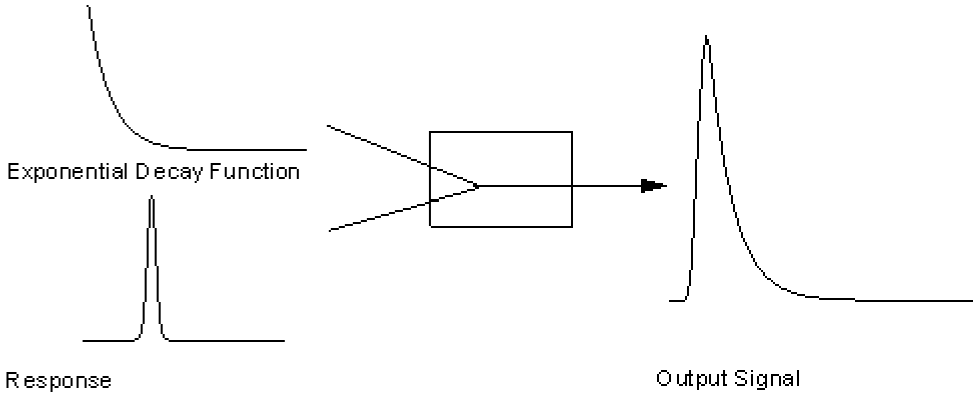
Background

Let's start this example by importing *Samples**Curve Fitting**FitConv.dat*.



A(X)	B(Y)	C(Y)
Sampling	Signal	Impulse
0	-0.19775	0
0.1	-0.32893	0
0.2	0.10055	0
0.3	0.09394	0
0.4	-0.1292	0
0.5	0.06346	1.48672E-6
0.6	0.19453	1.3383E-4

The source data includes sampling points, output signal and the impulse response. This experiment assumes that the output signal was the convolution of an exponential decay function with a Gaussian response:



Now that we already have the output signal and response data, we can get the exponential decay function by fitting the signal to the below model:

$$y = y_0 + \int_{-\infty}^{+\infty} Ae^{-tx} \otimes Response, dx$$

Define the Function

Obviously, column 1 and column 2 are x and y respectively in the function. How about column 3, the impulse response? We will access this column within the fitting function, and compute the theoretical exponential curve from the sampling points. Then we can use fast Fourier transform to perform the convolution.

Press **F9** to open the **Fitting Function Organizer** and define a function like:

Function Name:	FitConv
Function Type:	User-Defined
Independent Variables:	x
Dependent Variables:	y
Parameter Names:	y0, A, t
Function Form:	Origin C
Function:	

Click the button (icon) beside the **Function** box and write the function body in **Code Builder**:

```
#pragma warning(error : 15618)
#include <origin.h>
// Header files need to be included
#include <ONLSF.H>
#include <fft_utils.h>
//
//
void _nlsfTestConv(
// Fit Parameter(s):
double y0, double A, double t,
// Independent Variable(s):
double x,
// Dependent Variable(s):
double& y)
{
        // Beginning of editable part
        Worksheet wks = Project.ActiveLayer();
        NLFitContext *pCtxt = Project.GetNLFitContext();
        if ( pCtxt )
        {
                // Vector for the output signal in each iteration.
                static vector vSignal;
                // If parameters were updated, we will recalculate the convolution result.
                BOOL bIsNewParamValues = pCtxt->IsNewParamValues();
                if ( bIsNewParamValues )
                {
                        // Read sampling and response data from worksheet.
                        Dataset dsSampling(wks, 0);
                        Dataset dsResponse(wks, 2);
                        int iSize = dsSampling.GetSize();

                        vector vResponse, vSample;

                        vResponse = dsResponse;
                        vSample = dsSampling;

                        vSignal.SetSize(iSize);
                        vResponse.SetSize(iSize);
                        vSample.SetSize(iSize);

                        // Compute the exponential decay curve
                        vSignal = A * exp( -t*vSample );
                        // Perform convolution
                        int iRet = fft_fft_convolution(iSize, vSignal, vResponse);

                }

                NLSFCURRINFO    stCurrInfo;
                pCtxt->GetFitCurrInfo(&stCurrInfo);
                // Get the data index for the iteration
                int nCurrentIndex = stCurrInfo.nCurrDataIndex;
                // Get the evaluated y value
                y = vSignal[nCurrentIndex] + y0;
                // For compile the function, since we haven't use x here.
                x;
        }
        // End of editable part
}
```

Traditionally, for a particular x, the function will return the corresponding y value. However, when convolution is involved, we need to perform the operation on the entire curve, not only on a particular data point. So from Origin 8 SR2, we introduced the NLFitContext class to achieve some key information within the fitter. In each iteration, we use NLFitContext to monitor the fitted parameters; once they are updated, we will compute the convolution using the fast Fourier transform by the fft_fft_convolution method. The results are saved in the vSignal vector. Then for each x, we can get the evaluated y from vSignal with the current data index in NLSFCURRINFO.

Fit the Curve

In the fitting function body, we read the response data directly from the active worksheet. So you should perform the fit from the worksheet. Highlight column B and press **Ctrl + Y** to bring up the **Nonlinear Fitting** dialog. Select the *FitConv* function, and then initialize the parameters as y0=0, A=10, t=1. Click the **Fit** button to generate the results.

7.14 Quoting Built-in Functions in Your New Function

7.14.1 Summary

This tutorial will show you how to reference an built-in function when creating a user-defined fitting function.

Minimum Origin Version Required: Origin 8.0 SR6

7.14.2 What you will learn

This tutorial will show you how to:

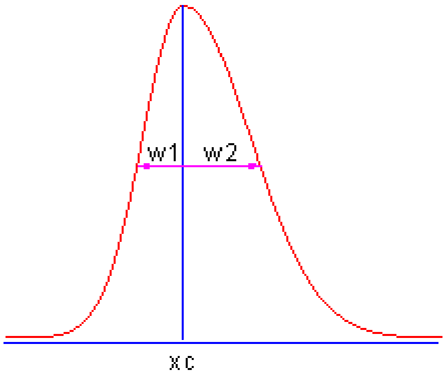
- Define a piecewise fitting function
- Access built-in functions in you new function
- Auto initialize the parameters

7.14.3 Steps

Data

Start by import the file *Samples**Curve Fitting**Asymmetric Gaussian.dat* into a new workbook.

Highlight column B and create a graph. The peak in the data is slightly skewed to the right. How to fit such a curve? One idea is to divide the curve into two parts - We can consider this curve to be composed of two Gaussian function as below. These two Gaussian curves share the same baseline and peak center, but different with the peak width and amplitude.



Define the Function

Press **F9** to open the **Fitting Function Organizer** and define the function as below:

Function Name:	AsymmetricGauss
Function Type:	User-Defined
Independent Variables:	x
Dependent Variables:	y
Parameter Names:	y0, xc, w1, w2, A1, A2
Function Form:	Origin C
Function:	y = x<xc? nlf_Gauss(x, y0, xc, w1, A1) : nlf_Gauss(x, y0, xc, w2, A2);

Note:

For versions before **Origin 8.1**, the function body should be defined as:

```
y = x<xc? nlfxGauss(x, y0, xc, w1, A1) : nlfxGauss(x, y0, xc, w2, A2);
x; y0; xc; w1; w2; A1; A2;
```

Listing the parameters at the end is used to avoid the "parameter not used inside the function body" error, although you already use these parameters. This is required to compile the function successfully.

When calling **nlf_*FuncName*** to reuse built-in functions, the syntax is:

```
        nlf_FuncName( independent variable, parameter list ... )
```

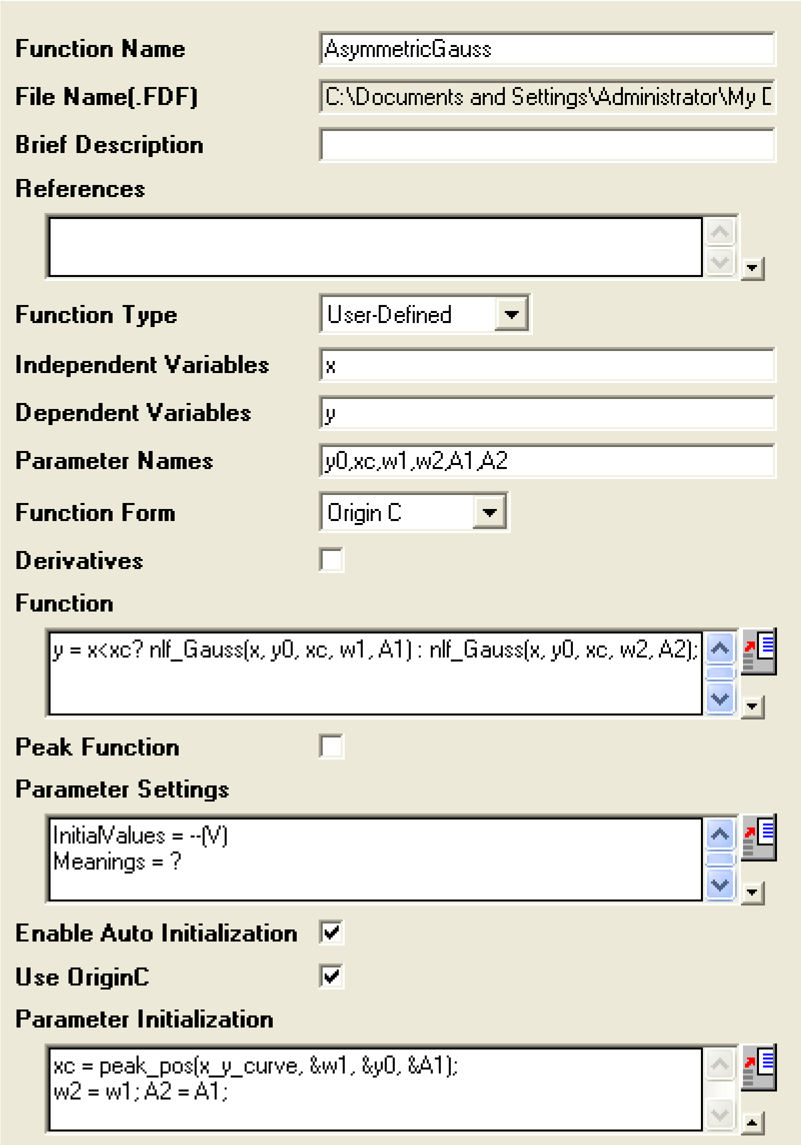
where ***FuncName*** is the fitting function name. Besides, the old notation, ***nlfxFuncName*** also supported.

The *Parameter List* follows the parameter order in function definition file for the built-in function (the FDF file. You can open the FDF file in Notepad. The files are located in the \\Origin EXE Folder\FitFunc\). Note that, the function name we use is the DLL interface name. The actual name in the [General Information] section of the FDF file. Look at the *Function Source* item and the value is **fgroup.*FuncName***, and use the *FuncName*. In most cases, this function name is consistent with the function name visible in the NLFit dialog. For a few few functions such as Voigt, these names are different.

For parameter initialization of this skewed gaussian function, we can simply copy the initialization code of the built-in gauss function, and make a few minor modifications:

```
xc = peak_pos(x_y_curve, &w1, &y0, &A1);
w2 = w1;
A2 = A1;
```

The final function body should be as below:



Function Name	AsymmetricGauss
File Name(.FDF)	C:\Documents and Settings\Administrator\My D
Brief Description	
References	
Function Type	User-Defined
Independent Variables	x
Dependent Variables	y
Parameter Names	y0,xc,w1,w2,A1,A2
Function Form	Origin C
Derivatives	□

Function

```
y = x<xc? nlf_Gauss(x, y0, xc, w1, A1) : nlf_Gauss(x, y0, xc, w2, A2);
```

Peak Function □

Parameter Settings

```
InitialValues = --(V)
Meanings = ?
```

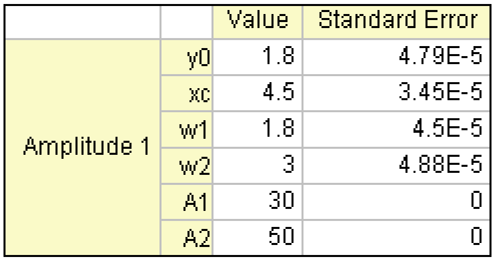
Enable Auto Initialization ☑

Use OriginC ☑

Parameter Initialization

```
xc = peak_pos(x_y_curve, &w1, &y0, &A1);
w2 = w1; A2 = A1;
```

Once compiled successfully, save the function and fit the curve. The results should be as below:



		Value	Standard Error
Amplitude 1	y0	1.8	4.79E-5
	xc	4.5	3.45E-5
	w1	1.8	4.5E-5
	w2	3	4.88E-5
	A1	30	0
	A2	50	0

7.15 Fit Function with Non-constant Background

7.15.1 Summary

Many of the Origin built-in functions are defined as:

$$y = y_0 + ...$$

Where y0 can be treated as the "constant background". How about fitting a curve with a non-constant background? One option is to use the Peak Analyzer we provide. The Peak Analyzer includes multiple methods to subtract the baseline, including exponential or polynomial backgrounds. In this tutorial, we will show you how to fit such curves without using the Peak Analyzer.

Minimum Origin Version Required: Origin 8.0 SR6

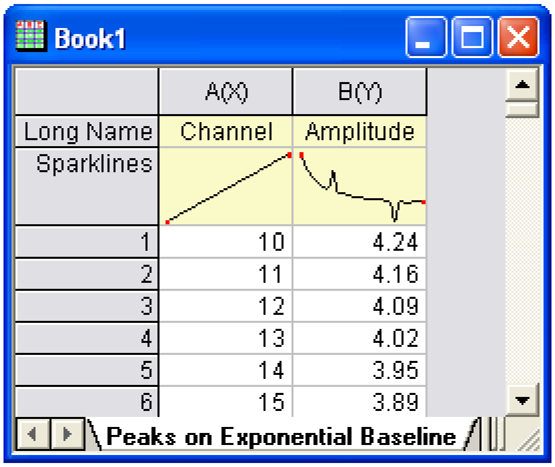
7.15.2 What you will learn

- Review Extract Worksheet Data.
- Quote a built-in function by nlfx*FuncName* method.
- Auto Initialize the parameters.

7.15.3 Example and Steps

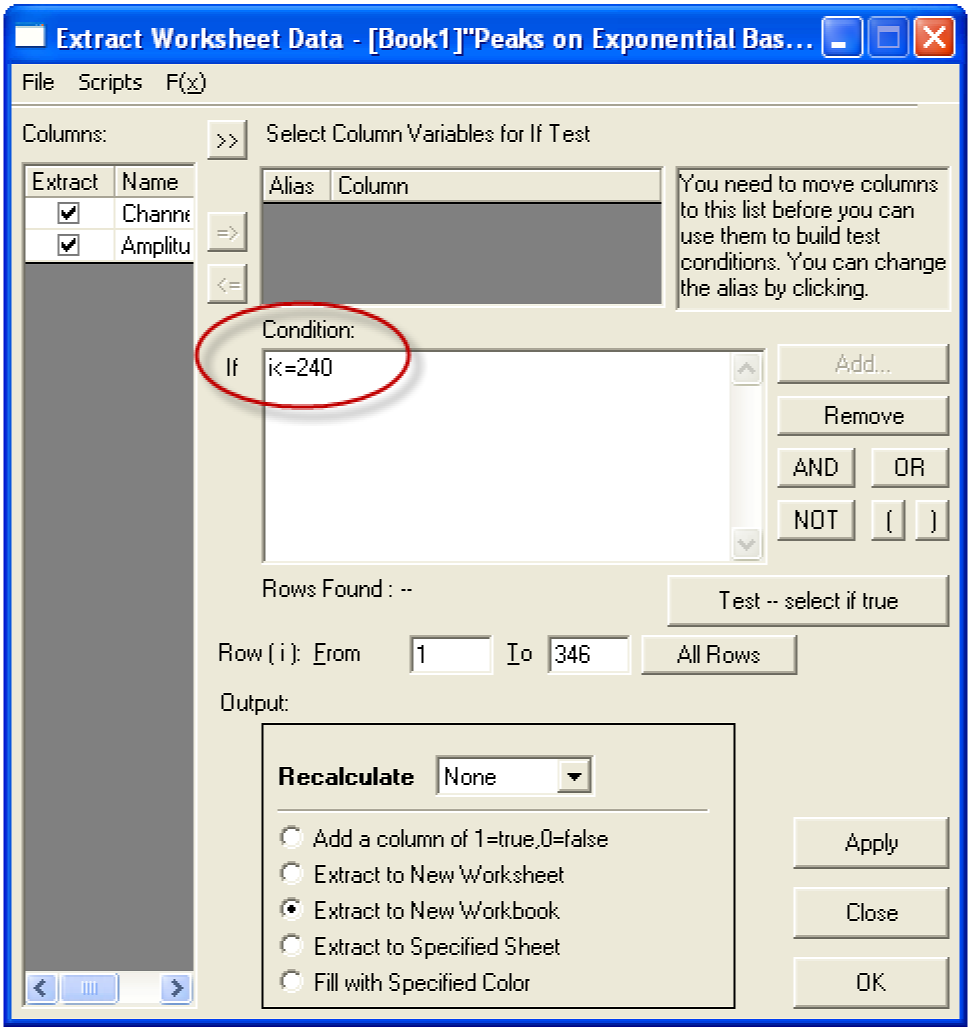
Prepare the Data

Let's start this tutorial by importing *\Samples\Spectroscopy\Peaks on Exponential Baseline.dat*. From the worksheet sparkline, we can see that there are two peaks in the curve. To simplify the problem, we will fit just one peak in this example.



	A(X)	B(Y)
Long Name	Channel	Amplitude
Sparklines		
1	10	4.24
2	11	4.16
3	12	4.09
4	13	4.02
5	14	3.95
6	15	3.89

Now bring up the Extract Worksheet Data dialog from **Worksheet : Extract Worksheet Data**. And we will extract data from row 1 to row 240:



Extract Worksheet Data - [Book1]"Peaks on Exponential Bas...

Columns:

Extract	Name
☑	Channe
☑	Amplitu

Select Column Variables for If Test

Alias	Column

You need to move columns to this list before you can use them to build test conditions. You can change the alias by clicking.

Condition:

If i<=240

Rows Found : --

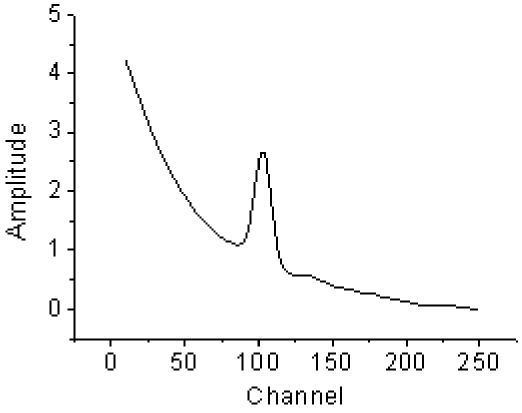
Row (i): From 1 To 346

Output:

Recalculate None

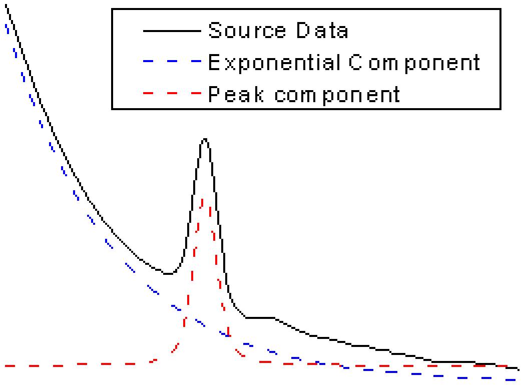
○ Add a column of 1=true,0=false
○ Extract to New Worksheet
● Extract to New Workbook
○ Extract to Specified Sheet
○ Fill with Specified Color

So the curve we will fit should look like this:



Define the Function

As illustrated below, we can consider the source curve is the combination of an exponential decay component (the background) with a Voigt peak:



So should we write down the whole equation to define the function? Like:

$$y = y_0 + A_1 e^{-x/t1} + A_2 \frac{2w_L \ln 2}{\pi^{\frac{3}{2}} w_G^2} \int_{-\infty}^{\infty} \frac{e^{-t^2}}{\left(\sqrt{\ln 2}\frac{w_L}{w_G}\right)^2 + \left(\sqrt{4\ln 2}\frac{x-x_c}{w_G} - t\right)^2} dt$$

Well, this is a complicated equation and it includes infinite integration. Writing such an equation directly is painful. Now that we already have these two built-in functions:

ExpDec1:

$$y = y_0 + Ae^{-x/t}$$

Voigt:

$$y = y_0 + A \frac{2w_L \ln 2}{\pi^{\frac{3}{2}} w_G^2} \int_{-\infty}^{\infty} \frac{e^{-t^2}}{\left(\sqrt{\ln 2}\frac{w_L}{w_G}\right)^2 + \left(\sqrt{4\ln 2}\frac{x-x_c}{w_G} - t\right)^2} dt$$

we can simply use the **nlfx*FuncName*** method to quote these two built-in functions and create a new one. Press **F9** to open the **Fitting Function Organizer** and define a function as below:

Function Name:	ExpVoigt
Function Type:	User-Defined
Independent Variables:	x
Dependent Variables:	y
Parameter Names:	y0, A1, t1, xc, A2, wG, wL
Function Form:	Origin C
Function:	y = nlf_ExpDec1(x, y0, A1, t1) + nlf_Voigt(x, y0, xc, A2, wG, wL) - y0;

Note:

Some of the built-in function names do not consistent with the actual DLL function name. Just like this Voigt function, it's defined in Voigt5.FDF, and if you open the FDF file by Notepad, you can see a line under [GENERAL INFORMATION] section says:

Function Source=fgroup.Voigt5

The name after "fgroup" is the actual name we should put into **nlf_*FuncName***.

Besides, for versions before **Origin 8.1 SR2**, the function body should use old *nlfxFuncName* notation and define as:

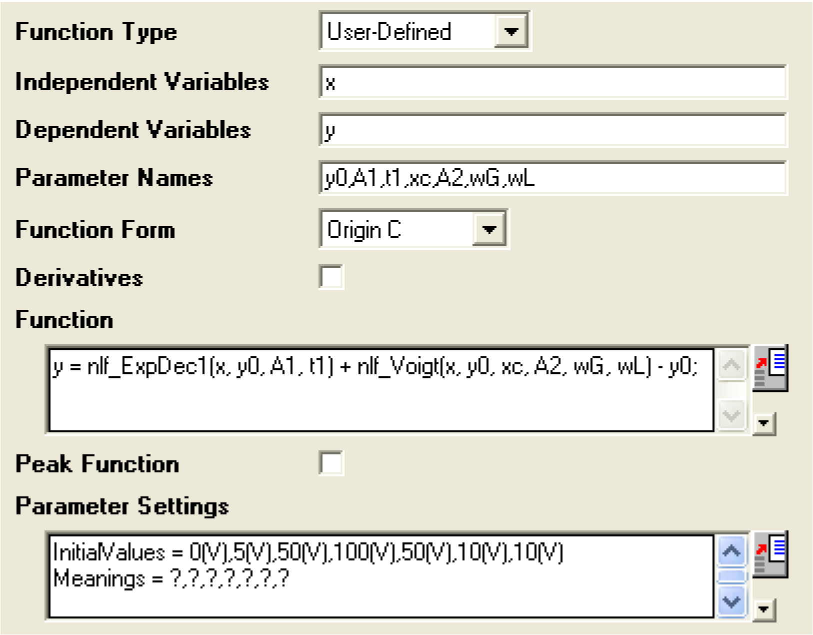
```
y = nlfxExpDec1(x, y0, A1, t1) + nlfxVoigt(x, y0, xc, A2, wG, wL) - y0;
x; xc; A1; t1; A2; wG; wL;
```

Listing the parameters at the end is done to avoid the "parameter not used inside the function body" error, although you already use these parameters. If not, you will not compile the function successfully.

Click the button on the right of the **Parameter Settings** and enter these parameter initial values:

y0:	0
A1:	5
t1:	50
xc:	100
A2:	50
wG:	10
wL:	10

So the final function definition part should look like:



Function Type	User-Defined
Independent Variables	x
Dependent Variables	y
Parameter Names	y0,A1,t1,xc,A2,wG,wL
Function Form	Origin C
Derivatives	□
Function	
Peak Function	□
Parameter Settings	

Function:

```
y = nlf_ExpDec1(x, y0, A1, t1) + nlf_Voigt(x, y0, xc, A2, wG, wL) - y0;
```

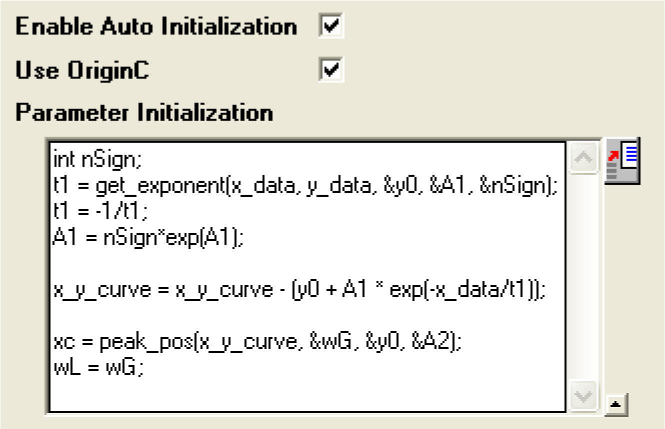
Parameter Settings:

```
InitialValues = 0(V),5(V),50(V),100(V),50(V),10(V),10(V)
Meanings = ?,?,?,?,?,?,?
```

Auto Parameter Initialization

In the above section, we set fixed parameter initial values. If you know the possible fitted results, you can set the initial values in this way. But how about when the data is changed? Origin provides an Origin C interface to "guess" the initial values. To use the parameter initialization code, make sure to check the **Enable Auto Initialization** and **Use OriginC** checkboxes, and edit the code in Code Builder by clicking the icon.

(P.S: If you know the initial values very well, or you don't like coding, please skip this section.)



Enable Auto Initialization ☑

Use OriginC ☑

Parameter Initialization

```
int nSign;
t1 = get_exponent(x_data, y_data, &y0, &A1, &nSign);
t1 = -1/t1;
A1 = nSign*exp(A1);

x_y_curve = x_y_curve - (y0 + A1 * exp(-x_data/t1));

xc = peak_pos(x_y_curve, &wG, &y0, &A2);
wL = wG;
```

Now that the curve is composed by two components, we can guess the parameter values by separating these two parts, the initialization code includes:

1. Use the get_exponent function to fit the curve and get the parameter values for exponential component.
2. Remove the background -- exponential component -- from source data.
3. Approaching the peak by Gaussian peak using peak_pos function and set the initial values for peak component

So, the initialization code in Code Builder should look like this:

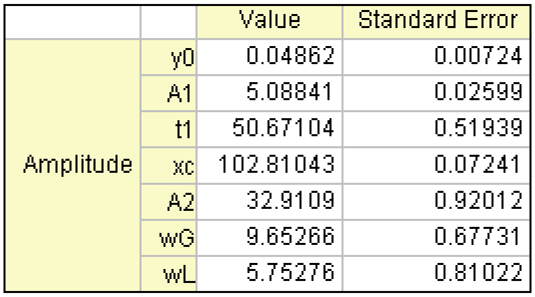
```
void _nlsfParamExpVoigt(
// Fit Parameter(s):
double& y0, double& A1, double& t1, double& xc, double& A2, double& wG, double& wL,
// Independent Dataset(s):
vector& x_data,
// Dependent Dataset(s):
vector& y_data,
// Curve(s):
Curve x_y_curve,
// Auxilary error code:
int& nErr)
{
        // Beginning of editable part
        int nSign;
        // Evaluates the parameters' value, y0, ln(A) and R for y = y0+A*exp(R*x).
        t1 = get_exponent(x_data, y_data, &y0, &A1, &nSign);
        // Set the exponential component values for the fitting function.
        t1 = -1/t1;
        A1 = nSign*exp(A1);
        // Remove the exponential component from the curve;
        x_y_curve = x_y_curve - (y0 + A1 * exp(-x_data/t1));
        // Fit to get peak values.
        xc = peak_pos(x_y_curve, &wG, &y0, &A2);
        wL = wG;
        // End of editable part
}
```

Note:

When you check the **Enable Auto Initialization** and enter the initialization code, this code will cover the initial values in **Parameter Settings**.

Fit the Curve

No matter what kind of parameter initialization method you used, highlight column B and press **Ctrl + Y** to bring up the NLFit dialog, select the ExpVoigt function and fit. The result should be:



		Value	Standard Error
Amplitude	y0	0.04862	0.00724
	A1	5.08841	0.02599
	t1	50.67104	0.51939
	xc	102.81043	0.07241
	A2	32.9109	0.92012
	wG	9.65266	0.67731
	wL	5.75276	0.81022

7.16 Fitting with Piecewise Functions

7.16.1 Summary

We will show you how to define piecewise fitting function in this tutorial.

Minimum Origin Version Required: Origin 8.0 SR6

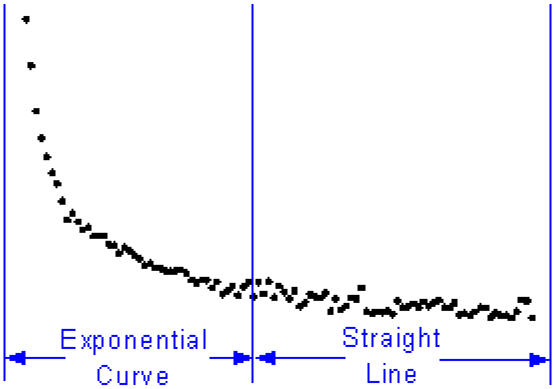
7.16.2 What you will learn

This tutorial will show you how to:

- Define piecewise (conditional) fitting functions.

7.16.3 Example and Steps

We can start this tutorial by importing the sample *Samples**Curve Fitting**Exponential Decay*.*dat* data file. Highlight column D and plot a Scatter Graph. You can fit this curve using built-in functions under Growth/Sigmoidal category, however, in this tutorial, we will separate the curve into two parts by a piecewise function.



So the equation will be:

$$y = \begin{cases} a + bx + e^{-\frac{x - x_c}{t1}}, & \text{if } x < x_c \\ a + bx, & \text{if } x \geq x_c \end{cases}$$

Define the Function

Press **F9** to open the **Fitting Function Organizer** and define a function like:

Function Name:	piecewise
Function Type:	User-Defined
Independent Variables:	x
Dependent Variables:	y
Parameter Names:	xc, a, b, t1
Function Form:	Origin C
Function:	

Click the button on the right of the **Function** edit box and define the fitting function in Code Builder using:

```
void _nlsfpiecewise(
// Fit Parameter(s):
double xc, double a, double b, double t1,
// Independent Variable(s):
double x,
// Dependent Variable(s):
double& y)
{
        // Beginning of editable part
        // Divide the curve by if condition.
        if(x<xc) {
                y = a+b*x+exp(-(x-xc)/t1);
        } else {
                y = a+b*x;
        }
        // End of editable part
}
```

Fit the Curve

Press **Ctrl + Y** to bring up NLFit dialog with the graph window active. Select the *piecewise* function we defined and initialize the parameter values:

xc:	1
a:	1
b:	-1
t1:	0.1

Click **Fit** button to generate the results:

xc:	0.24
a:	36.76585
b:	-24.62876
t1:	0.04961

Note that this function is sensitive to *xc* and *t1*, different initial values could generate different results.

8 Peaks and Baseline

- Picking and Marking Peaks
- Integrating Peaks
- Peak Fitting with Baseline
- Peak Fitting with Preset Peak Parameters
- Setting the Fix, Share Status or Bounds for Multiple Peak Parameters Simultaneously

8.1 Picking and Marking Peaks

8.1.1 Summary

The Peak Analyzer provides several methods to pick peaks automatically. Also, user can opt to add/delete/modify the peaks manually.

Labels are added to the peak centers after they are found or added, to show user the positions of the current peaks.

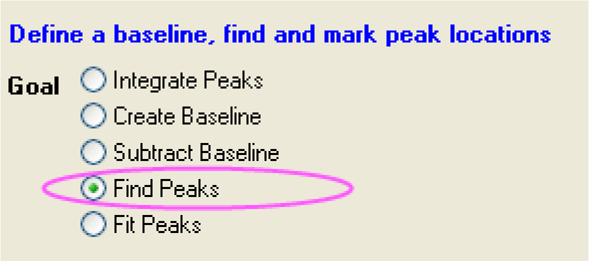
Minimum Origin Version Required: Origin 8.0 SR6

8.1.2 What you will learn

- How to use and customize the auto peak finding in the Peak analyzer
- How to customize the labels for the peak centers

8.1.3 Steps

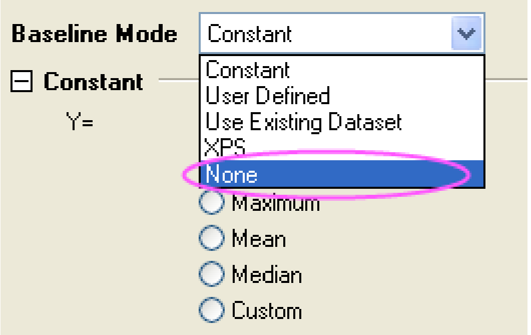
1. Start a new workbook and import the file <Origin Program Folder>\Samples\Spectroscopy\HiddenPeaks.dat.
2. Highlight the second column.
3. Create a line plot by selecting **Plot: Line: Line**.
4. With the graph active, select **Analysis: Peaks and Baseline: Peak Analyzer** to open the dialog of the Peak Analyzer.
5. In the first page (the **Start** page), select the **Find Peaks** radio button in the **Goal** group. Then click the **Next** button to go to the next page.



Define a baseline, find and mark peak locations

Goal
○ Integrate Peaks
○ Create Baseline
○ Subtract Baseline
● Find Peaks
○ Fit Peaks

6. In the **Baseline Mode** page, select **None** for **Baseline Mode**.



Baseline Mode: Constant

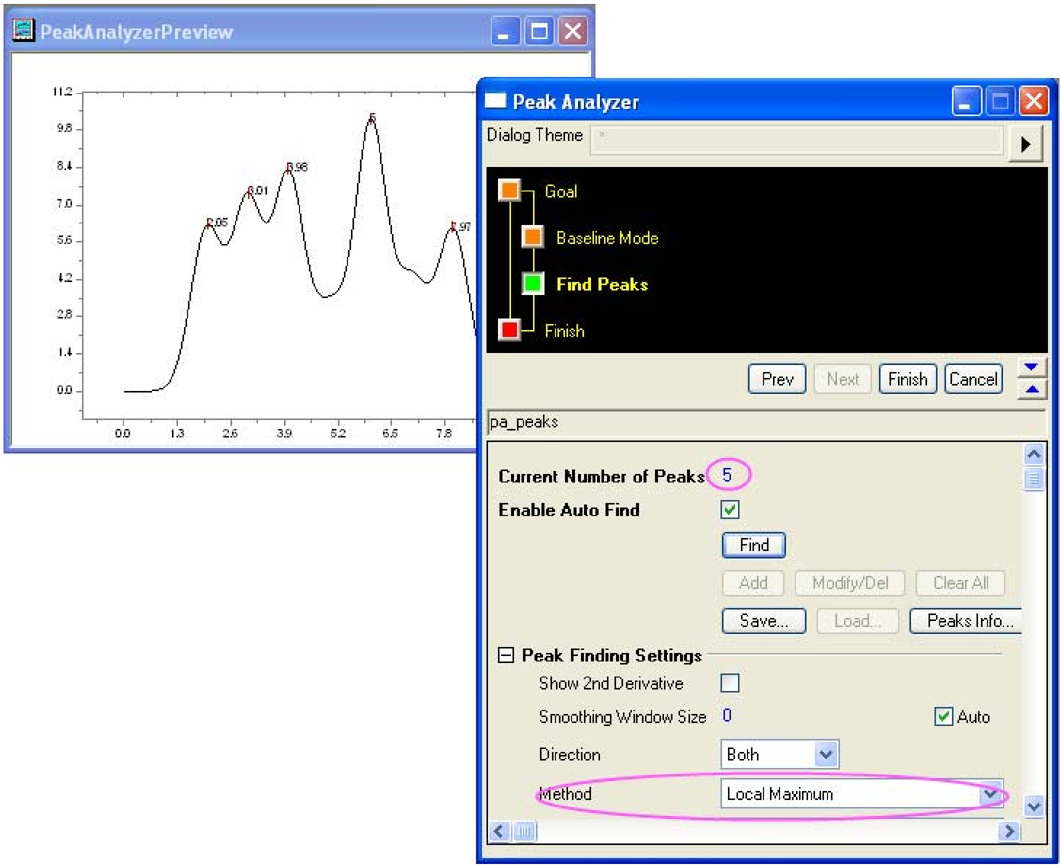
⊟ Constant

Y=

Click the **Next** button to go to the **Find Peaks** page.

7. In the **find Peaks** page:

    1. Expand the **Peak Finding Settings** branch. Make sure that **Local Maximum** is selected for **Method**. Then click the **Find** button. Only five peaks are detected.



Dialog Theme

pa_peaks

Current Number of Peaks: 5

Enable Auto Find: ☑

⊟ Peak Finding Settings

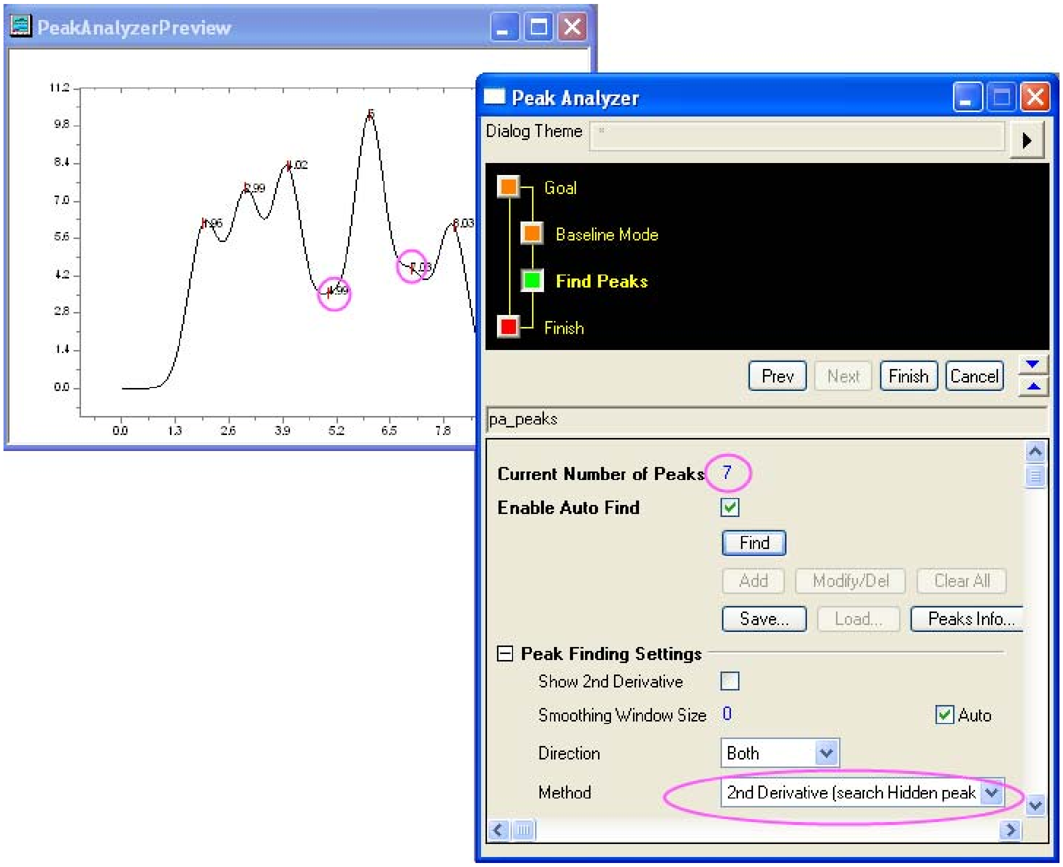
Show 2nd Derivative: □

Smoothing Window Size: 0 ☑ Auto

Direction: Both

Method: Local Maximum

    2. Change **Method** to **2nd Derivative (Search Hidden Peaks)**. Click the **Find** button again. This time, seven peaks are detected.



Peak Analyzer

Dialog Theme

Goal

Baseline Mode

Find Peaks

Finish

Prev Next Finish Cancel

pa_peaks

Current Number of Peaks 7

Enable Auto Find ☑

Find

Add Modify/Del Clear All

Save... Load... Peaks Info...

⊟ **Peak Finding Settings**

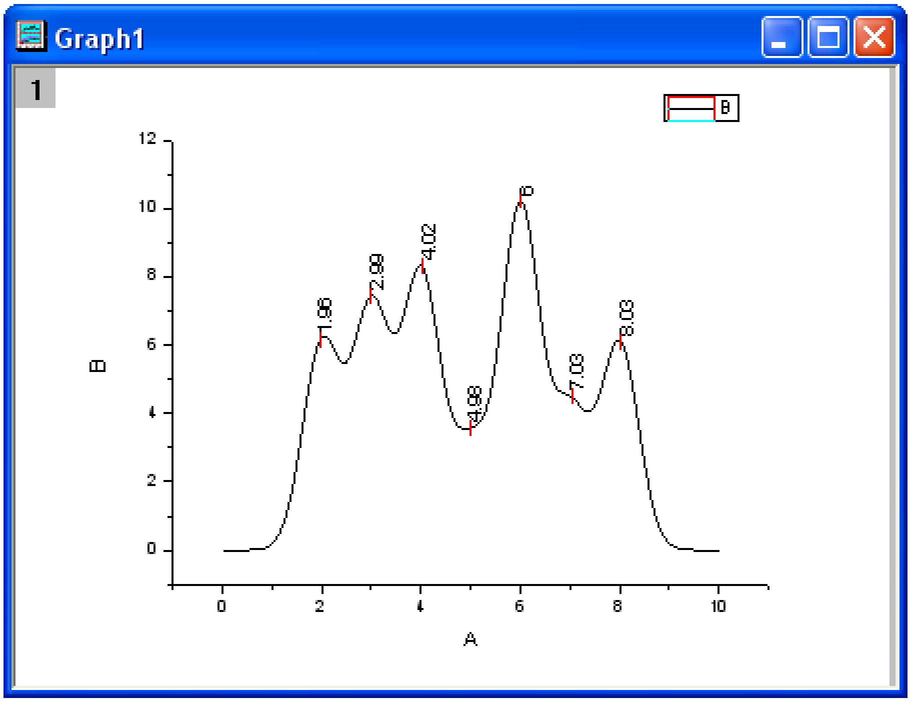
Show 2nd Derivative □

Smoothing Window Size 0 ☑ Auto

Direction Both

Method 2nd Derivative (search Hidden peak

3. Click **Finish** to complete the analysis. We will get this final graph:



8.2 Integrating Peaks

8.2.1 Summary

The Peak Analyzer is capable of integrating peaks to find their areas.

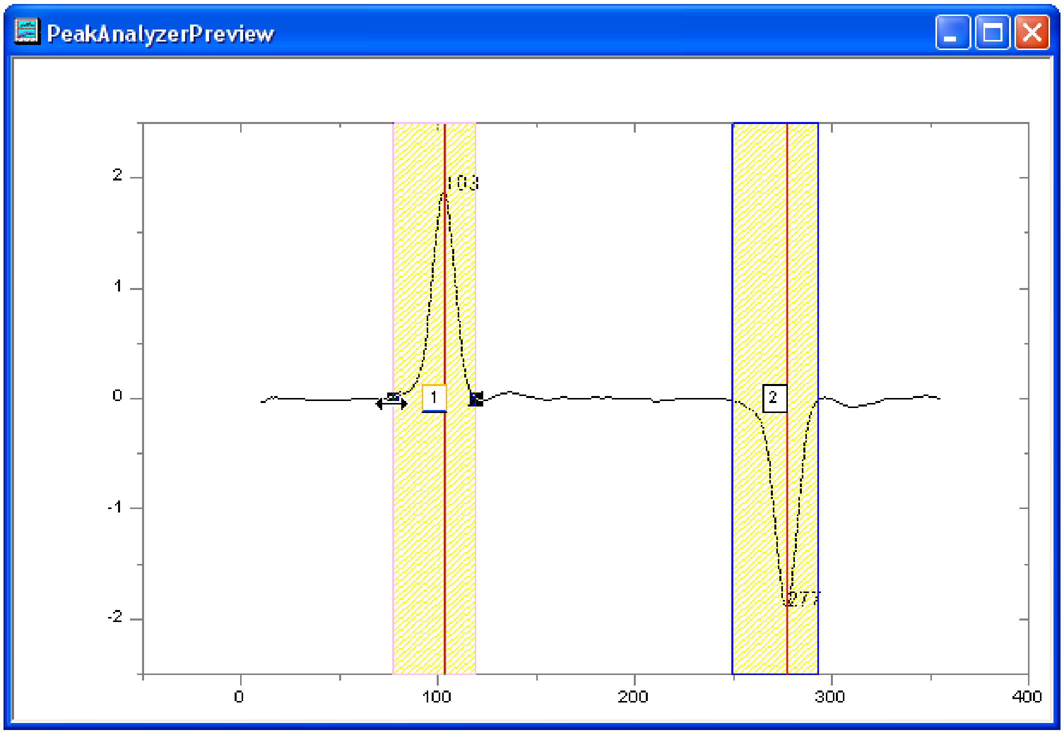
Minimum Origin Version Required: Origin 8.0 SR6

8.2.2 What you will learn

- How to pick an existing dataset as baseline
- How to subtract a baseline from the spectrum data
- How to calculate peak areas with the Peak Analyzer

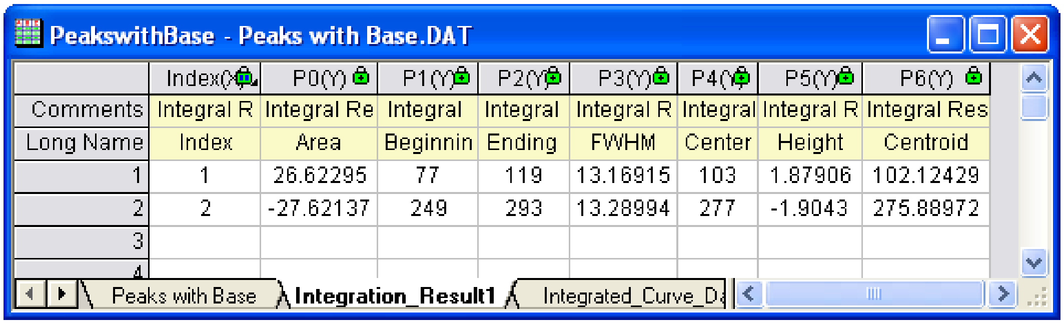
8.2.3 Steps

1. Start a new workbook and import the file *\Samples\Spectroscopy\Peaks with Base.DAT*.
2. Highlight the second column.
3. Select **Analysis: Peaks and Baseline: Peak Analyzer** from the main menu to open the **Peak Analyzer**.
4. In the first page (the **Start** page), select the **Integrate Peaks** radio button in the **Goal** group. Click **Next** to go to the **Baseline Mode** page.
5. In the **Baseline Mode** page, select **Use Existing Dataset** with the **Baseline Mode** drop-down list. Click the triangular button which next to the **Use Existing Dataset** to get the context menu and then select **Select Columns** to open the **Dataset Browser** to select the baseline dataset. In the **Dataset Browser** dialog, select column **C**, click **Add** button and then **OK** button to select the dataset as the baseline dataset. Click **Next** to go to the **Baseline Treatment** page.
6. Select the **Auto Subtract Baseline** check box. Click the page icon for the **Integrate Peaks** page in the wizard map to go to the last page (or you can click the **Next** button twice to directly go to the **Integrate Peaks** page). Please note that two peaks should be found in the **Find Peaks** page by using the default settings. You can see two numbered yellow rectangles added to them on the preview graph.
7. Set the **Integration Window Width** item as **Adjust on Preview Graph**, then click inside the rectangle marked with "1". A pair of handles (small black rectangles) appears on the left and right sides, allowing you to resize the yellow rectangle so as to change the range to perform integration on the first peak.



Similarly, you can resize the yellow rectangle around the second peak to modify the integration range for this peak.

8. In the last page of the Peak Analyzer, make sure all the desired quantities to compute have been selected in the **Quantities** group. For example, if you want to calculate the peak centroid for each peak, select the **Peak Centroid** check box. If you don't want to output the percent areas, clear the **Percent Area** check box. When you are done, click **Finish** to perform the analysis. The result is in a worksheet named *Integration_Result1*.



	Index(X)	P0(Y)	P1(Y)	P2(Y)	P3(Y)	P4(Y)	P5(Y)	P6(Y)
Comments	Integral R	Integral Re	Integral	Integral	Integral R	Integral	Integral R	Integral Res
Long Name	Index	Area	Beginnin	Ending	FWHM	Center	Height	Centroid
1	1	26.62295	77	119	13.16915	103	1.87906	102.12429
2	2	-27.62137	249	293	13.28994	277	-1.9043	275.88972
3								
4								

If the **Area** check box was selected in the **Quantities** group, you can see the peak areas in the **Area** column of this worksheet.

8.3 Peak Fitting with Baseline

8.3.1 Summary

In OriginPro, the **Peak Analyzer** is capable of performing multiple peak fitting with several baseline subtraction options.

There are various ways to create a baseline for your spectrum data. You can select a few anchor points and then fit them with a function. The fitting of the baseline can be done along with the peak fitting.

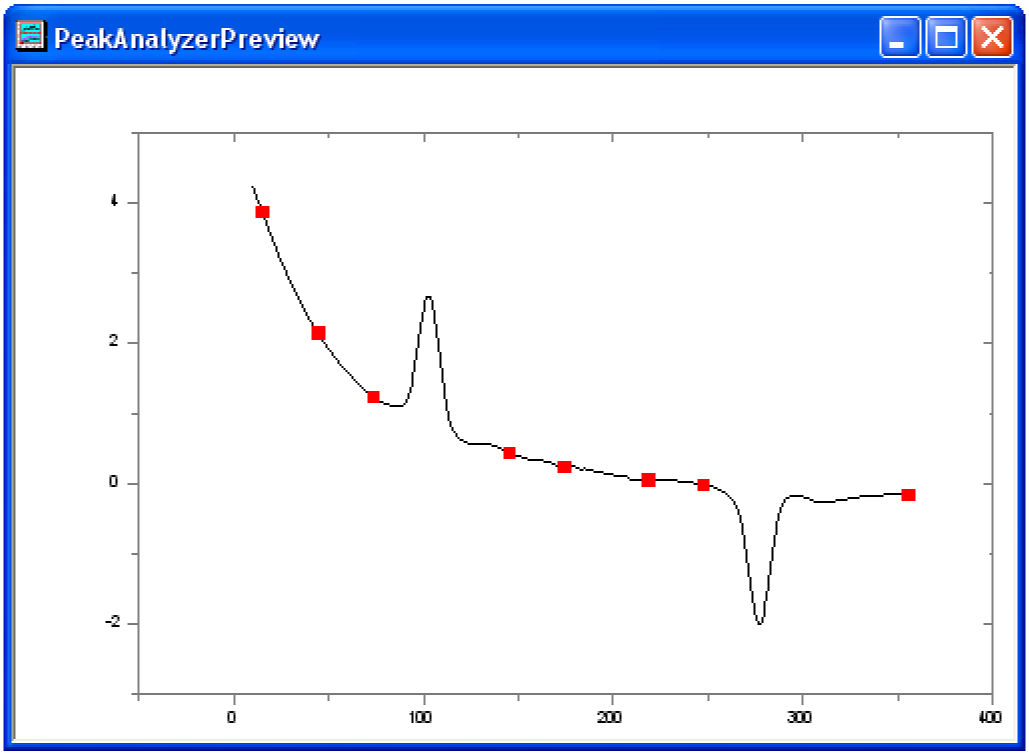
Minimum Origin Version Required: OriginPro 8.0 SR6

8.3.2 What You Will Learn

- How to perform fitting of peaks
- How to fit the baseline

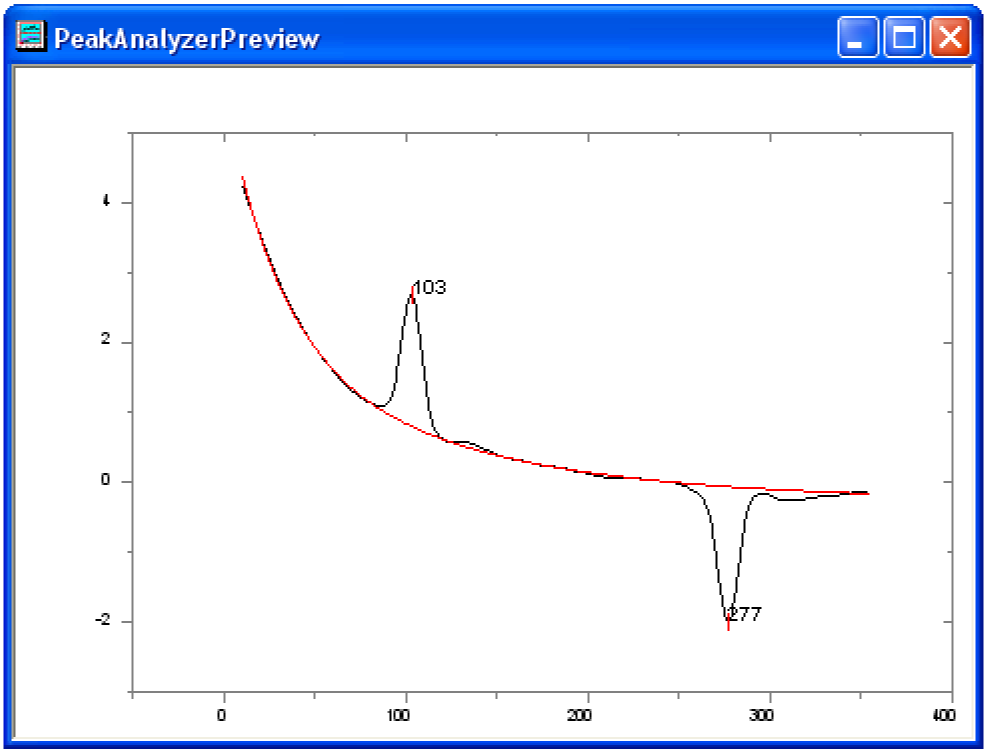
8.3.3 Steps

1. Start a new worksheet and import the file <Origin Program Folder>\Samples\Spectroscopy\Peaks on Exponential Baseline.dat.
2. Highlight the second column in the worksheet.
3. Select **Analysis: Peaks and Baseline: Peak Analyzer** from the main menu to open the **Peak Analyzer**.
4. Select the **Fit Peaks** radio button in the **Goal** group on the first page. Click **Next** to go to the **Baseline Mode** page.
5. In the **Baseline Mode** page, select **User Defined** from the **Baseline Mode** drop-down list. Click the **Find** button in the **Baseline Anchor Points** group. Eight anchor points should be found.



Click **Next** to go to the **Create Baseline** page.

6. In the **Create Baseline** page, select **Fitting** with the **Connect By** drop-down list. In the **Fitting** group, select **ExpDec2** from the **Function** drop-down list. Click **Next** to go to the **Baseline Treatment** page.
7. In the **Baseline Treatment** page, select the **Fit Baseline with Peaks** check box. Click **Next** to go to the **Find Peaks** page.
8. In the **Find Peaks** page, click the **Find** button to search peaks. Two peaks should be found.



Click **Next** to go to the **Fit Peaks** page.

9. In the **Fit Peaks** page, click the **Fit Control** button to open the **Peak Fit Parameters** dialog.
10. In the **Peak Fit Parameters** dialog, make sure that both peak types are Gaussian. Click the **Fit Until Converge** button. When the fitting is done, click **OK** to close the dialog.
11. Back in the **Fit Peaks** page, click **Finish** to complete the analysis. See the results in the source workbook and the graph report.

8.4 Peak Fitting with Preset Peak Parameters

8.4.1 Summary

In some cases, you may want to perform peak fitting with preset peak parameters. For example, you may have many datasets with fixed numbers of peaks and the centers of these peaks do not vary from dataset to dataset. What you are interested in is mainly other parameters of the peaks, for example, heights. Using the theme feature of the Peak Analyzer, you may carry out peak fitting with fixed peak parameters easily.

Minimum Origin Version Required: OriginPro 8.0 SR6

8.4.2 What You Will Learn

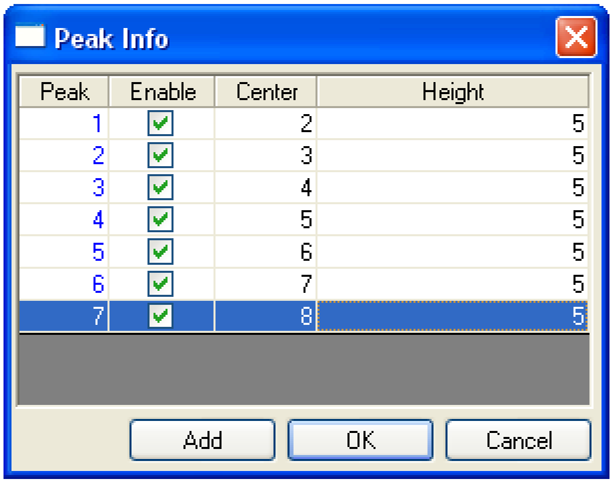
1. How to save Peak Analyzer settings in a theme and reuse them
2. How to fix peak parameters

8.4.3 Steps

Save a theme with peak positions and peak parameters

1. Start a new worksheet and import the file <Origin Program Folder>\Samples\Spectroscopy\HiddenPeaks.dat.
2. Highlight the second column and select **Analysis: Peaks and Baseline: Peak Analyzer** from the Origin menu to open the **Peak Analyzer**.
3. On the first page, select the **Fit Peaks** radio button in the **Goal** group. Click **Next** to go to the **Baseline Mode** page.
4. In the **Baseline Mode** page, select **None** with the **Baseline Mode** drop-down list. Click **Next** to go to the **Find Peaks** page.
5. In the **Find Peaks** page:
   a. Clear the **Enable Auto Find** check box, because we want to find the peaks manually. Click the **Peaks Info** button to open the **Peak Info** dialog.

   b. In the **Peak Info** dialog, click the **Add** button seven times to add 7 peaks. Enter the peak centers and heights as follows:



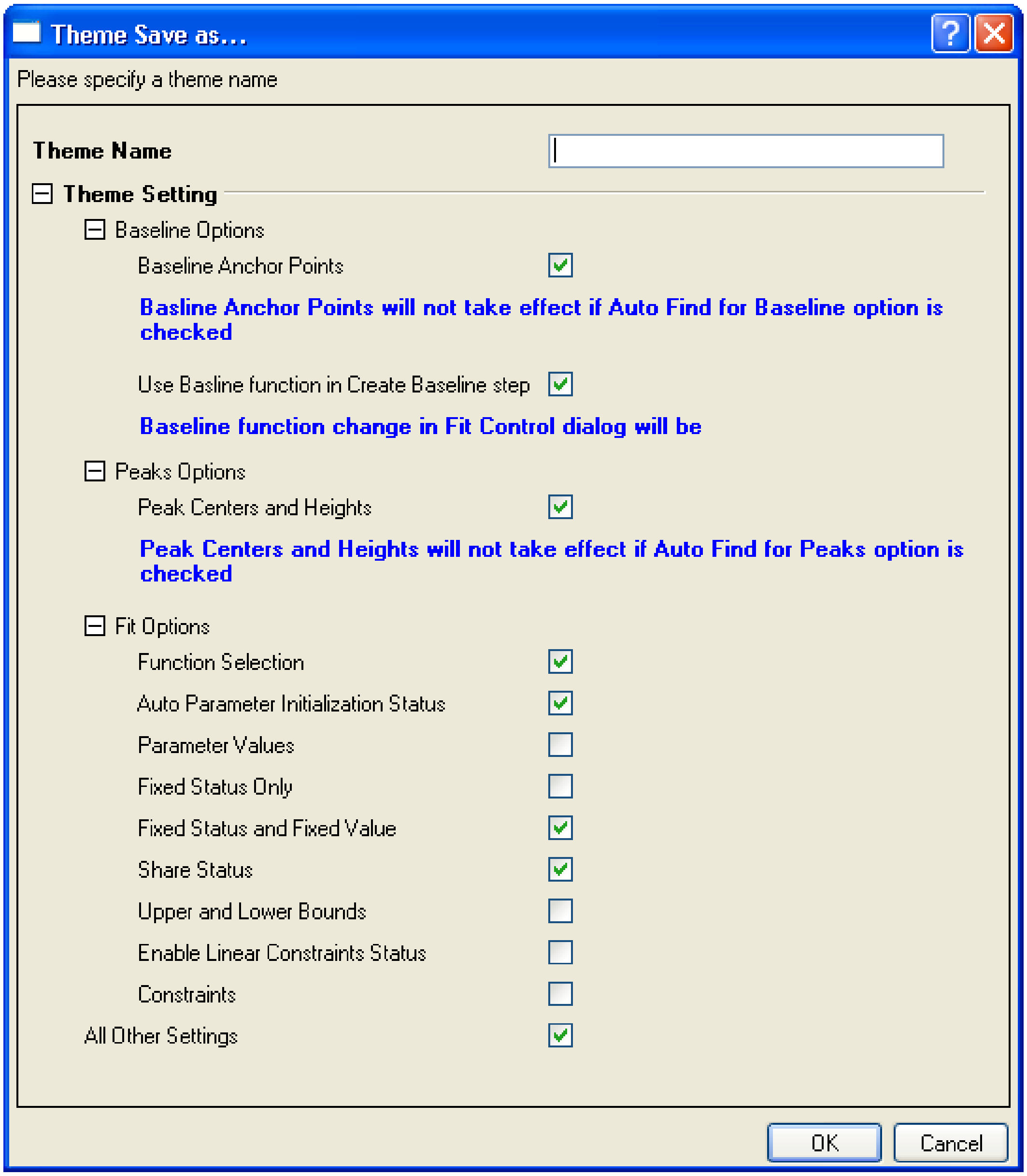
Peak	Enable	Center	Height
1	☑	2	5
2	☑	3	5
3	☑	4	5
4	☑	5	5
5	☑	6	5
6	☑	7	5
7	☑	8	5

   Click **OK** to return to the Peak Analyzer.

   c. Click **Next** to go to the **Fit Peaks** page.
6. In the **Fit Peaks** page:
   a. Click **Fit Control** to open the **Peak Fit Parameters** dialog.
   b. In the **Peak Fit Parameters** dialog, click the **Fix or release all peak centers** button. Then click the **Fit Until Converge** button. When the fitting is done, click **OK** to return to the **Peak Analyzer** dialog.

c. Click the right-sided triangle button to the right of **Dialog Theme** in the upper panel. Select **Save As** from the short-cut menu. The **Theme Save as** dialog opens.

d. In the **Theme Save as** dialog, enter **MyFitting** after **Theme Name**. Clear and select the check boxes as the screenshot below:



Please specify a theme name

Theme Name

⊟ **Theme Setting**

⊟ Baseline Options

Baseline Anchor Points ☑

Basline Anchor Points will not take effect if Auto Find for Baseline option is checked

Use Basline function in Create Baseline step ☑

Baseline function change in Fit Control dialog will be

⊟ Peaks Options

Peak Centers and Heights ☑

Peak Centers and Heights will not take effect if Auto Find for Peaks option is checked

⊟ Fit Options

Function Selection ☑

Auto Parameter Initialization Status ☑

Parameter Values ☐

Fixed Status Only ☐

Fixed Status and Fixed Value ☑

Share Status ☑

Upper and Lower Bounds ☐

Enable Linear Constraints Status ☐

Constraints ☐

All Other Settings ☑

Click **OK** to save the theme. This should bring you back to the **Peak Analyzer** dialog.

e. Click the **Finish** button in the **Peak Analyzer** to complete the analysis.

Reuse the theme

1. Start another new workbook and import the file <Origin Program Folder>\Samples\Spectroscopy\HiddenPeaks.dat.

2. Highlight the second column
3. Select **Analysis: Peaks and Baseline: Peak Analyzer** from the Origin menu to open the **Peak Analyzer** dialog.
4. On the first page of the **Peak Analyzer**, click the right-sided triangle button to the right of **Dialog Theme**. From the short-cut menu, pick **MyFitting**.
5. Click **Next** to check if the settings in every step are correct. Note that in the **Find Peaks** page, you can see the peak centers and heights are same as last time.
6. When you reach the last page, click the **Fit Control** button to open the **Peak Fit Parameters** dialog. Make sure that all peak centers are fixed and the values are the same as last time. Click **OK** to return to the **Peak Analyzer**.
7. Click **Finish** to complete the analysis. Check the results to see whether they are the same as the results we got last time.

8.5 Setting the Fix, Share Status or Bounds for Multiple Peak Parameters Simultaneously

8.5.1 Summary

When performing peak analysis, one often wants to fix parameter values, or share parameters between multiple peaks, or specify bounds. If your data has a few peaks, you can simply perform these settings for each peak. But if your data has many, it may be time consuming to set individually. To make the process more efficient, the **Peak Analyzer** offers context menus which can allow you to set the fix, share status or bounds for multiple peak parameters simultaneously. For more details about these settings, please refer to the Origin Help File.

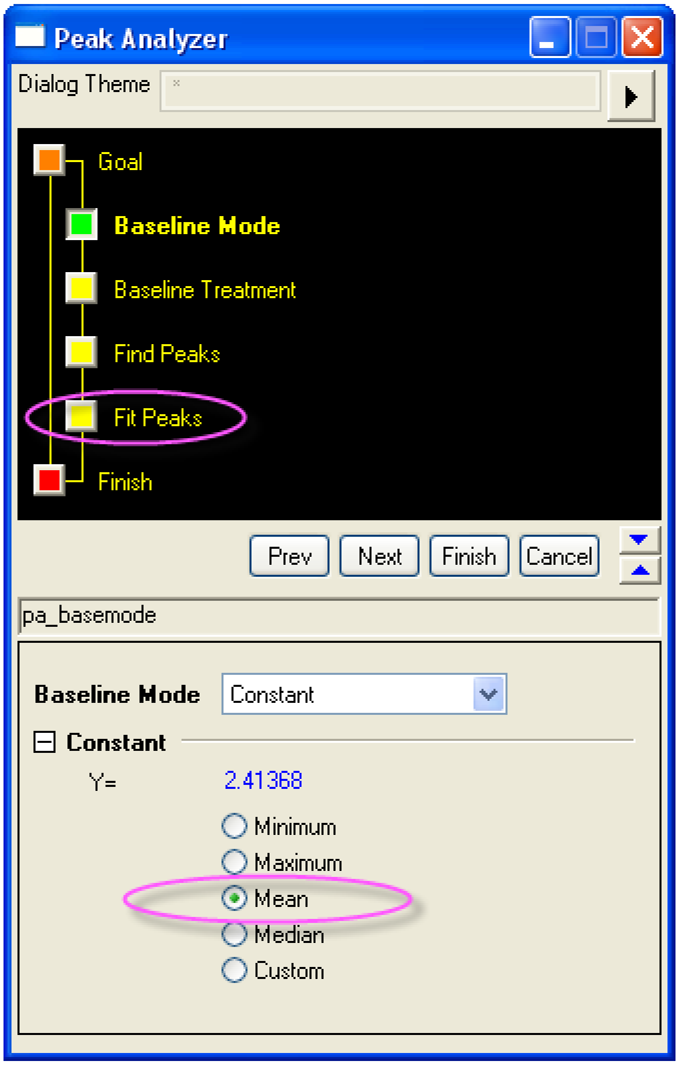
Minimum Origin Version Required: OriginPro 8.0 SR6

8.5.2 What You Will Learn

- How to set share status of multiple peak parameters simultaneously.
- How to set upper bounds and upper bound values to multiple peak parameters simultaneously.

8.5.3 Steps

1. Start a new workbook and import the file *<Origin Program Folder>\Samples\Spectroscopy\Positive & Negative Peaks.dat*.
2. Highlight the second column and select **Analysis: Peaks and Baseline: Peak Analyzer** to open the **Peak Analyzer** dialog. In the first page (the **Start** page), select the **Fit Peaks** radio button in the **Goal** group. Then press the **Next** button to go to the next page.
3. In the **Baseline Mode** page, select **Constant** with the **Baseline Mode** drop-down list and choose **Mean** in the **Constant** group. Then click **Fit Peaks** in the wizard map to directly go to the **Fit Peaks** page.



pa_basemode

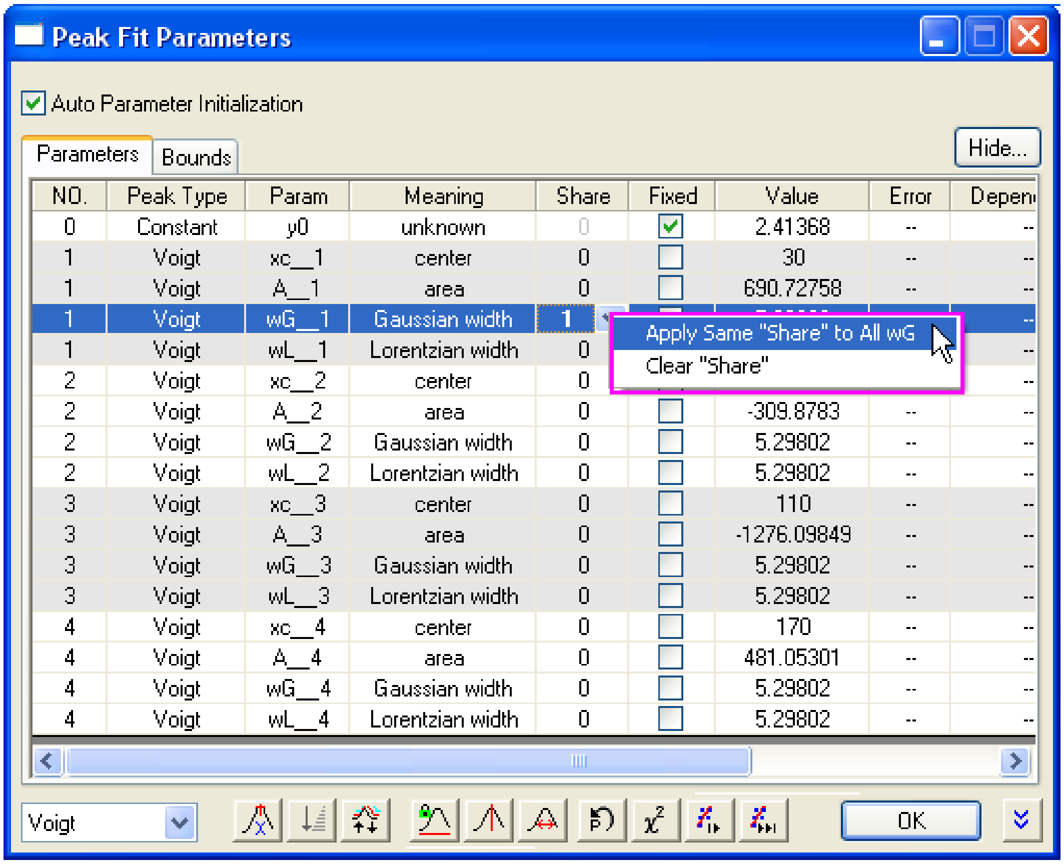
Baseline Mode Constant

Constant

Y= 2.41368

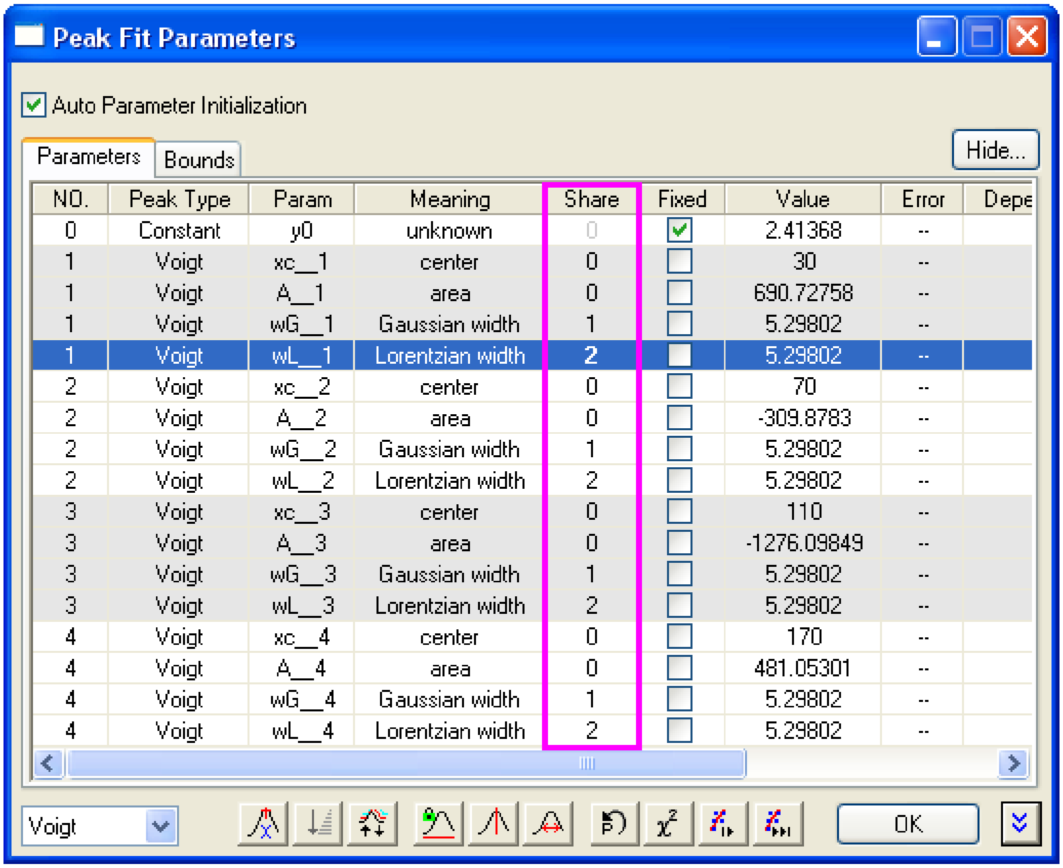
○ Minimum
○ Maximum
● Mean
○ Median
○ Custom

4. In the **Fit Peaks** page:
    1. click **Fit Control** button to open the **Peak Fit Parameters** dialog. In the lower left corner of the dialog, set the fitting function to **Voigt**.
    2. Make sure the **Parameters** tab is active and then select **1** in the **Share** column of the **wG_1** row. Then right click on it and select **Apply Same "Share" to All wG**. Then you will find that all the parameters with the **wG** prefix are shared in the same group.



NO.	Peak Type	Param	Meaning	Share	Fixed	Value	Error	Depen
0	Constant	y0	unknown	0	☑	2.41368	--	--
1	Voigt	xc__1	center	0	☐	30	--	--
1	Voigt	A__1	area	0	☐	690.72758	--	--
1	Voigt	wG__1	Gaussian width	1				--
1	Voigt	wL__1	Lorentzian width	0				--
2	Voigt	xc__2	center	0				--
2	Voigt	A__2	area	0	☐	-309.8783	--	--
2	Voigt	wG__2	Gaussian width	0	☐	5.29802	--	--
2	Voigt	wL__2	Lorentzian width	0	☐	5.29802	--	--
3	Voigt	xc__3	center	0	☐	110	--	--
3	Voigt	A__3	area	0	☐	-1276.09849	--	--
3	Voigt	wG__3	Gaussian width	0	☐	5.29802	--	--
3	Voigt	wL__3	Lorentzian width	0	☐	5.29802	--	--
4	Voigt	xc__4	center	0	☐	170	--	--
4	Voigt	A__4	area	0	☐	481.05301	--	--
4	Voigt	wG__4	Gaussian width	0	☐	5.29802	--	--
4	Voigt	wL__4	Lorentzian width	0	☐	5.29802	--	--

3. Then select **2** in the **Share** column of the **wL_1** row. Then right click on it and select **Apply Same "Share" to All wL**. Then you will find that all the parameters with the **wL** prefix are shared in the same group. After this, the **Parameters** tab should look like below:



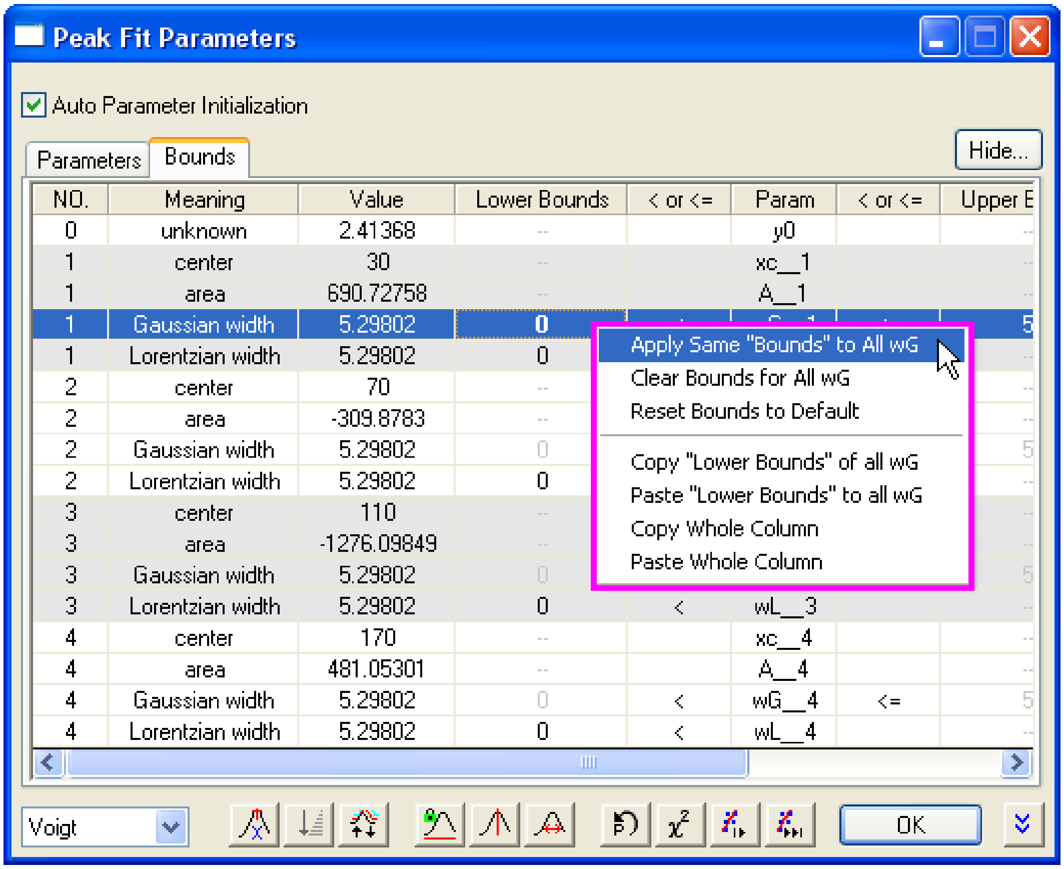
☑ Auto Parameter Initialization

NO.	Peak Type	Param	Meaning	Share	Fixed	Value	Error	Depe
0	Constant	y0	unknown	0	☑	2.41368	--	
1	Voigt	xc__1	center	0	☐	30	--	
1	Voigt	A__1	area	0	☐	690.72758	--	
1	Voigt	wG__1	Gaussian width	1	☐	5.29802	--	
1	Voigt	wL__1	Lorentzian width	2	☐	5.29802	--	
2	Voigt	xc__2	center	0	☐	70	--	
2	Voigt	A__2	area	0	☐	-309.8783	--	
2	Voigt	wG__2	Gaussian width	1	☐	5.29802	--	
2	Voigt	wL__2	Lorentzian width	2	☐	5.29802	--	
3	Voigt	xc__3	center	0	☐	110	--	
3	Voigt	A__3	area	0	☐	-1276.09849	--	
3	Voigt	wG__3	Gaussian width	1	☐	5.29802	--	
3	Voigt	wL__3	Lorentzian width	2	☐	5.29802	--	
4	Voigt	xc__4	center	0	☐	170	--	
4	Voigt	A__4	area	0	☐	481.05301	--	
4	Voigt	wG__4	Gaussian width	1	☐	5.29802	--	
4	Voigt	wL__4	Lorentzian width	2	☐	5.29802	--	

Voigt

4. Activate the **Bounds** tab. Double-click in the cell in the first **Gaussian width** row and the second **< or <=** column. And you will find **<=** is shown in this cell (the **<** will be shown if you double-click in the cell for one more time). Then type **5** into the **Upper Bounds** column in the same row.

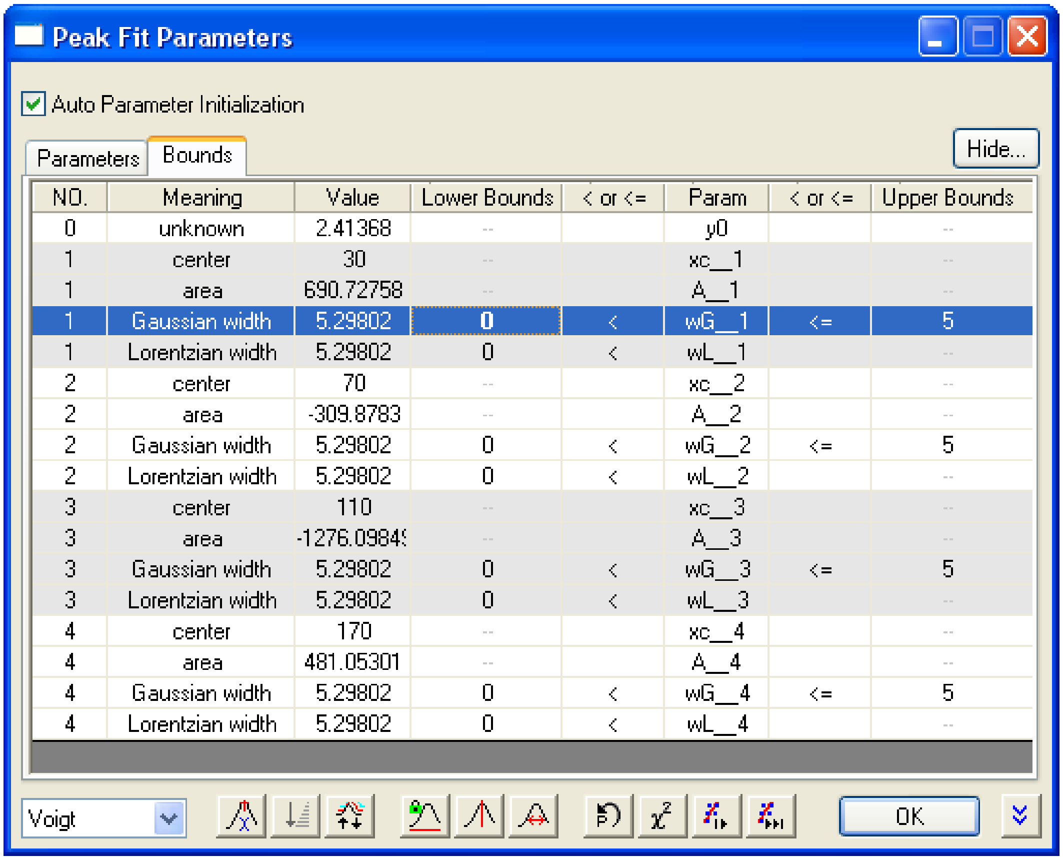
5. Then right click on it and select **Apply Same "Bounds" to All wG**.



☑ Auto Parameter Initialization

NO.	Meaning	Value	Lower Bounds	< or <=	Param	< or <=	Upper B
0	unknown	2.41368	--		y0		--
1	center	30	--		xc__1		--
1	area	690.72758	--		A__1		--
1	Gaussian width	5.29802	0				5
1	Lorentzian width	5.29802	0				--
2	center	70	--				--
2	area	-309.8783	--				--
2	Gaussian width	5.29802	0				5
2	Lorentzian width	5.29802	0				--
3	center	110	--				--
3	area	-1276.09849	--				--
3	Gaussian width	5.29802	0				5
3	Lorentzian width	5.29802	0	<	wL__3		--
4	center	170	--		xc__4		--
4	area	481.05301	--		A__4		--
4	Gaussian width	5.29802	0	<	wG__4	<=	5
4	Lorentzian width	5.29802	0	<	wL__4		--

After this, the **Bounds** tab should look like:



Peak Fit Parameters

☑ Auto Parameter Initialization

NO.	Meaning	Value	Lower Bounds	< or <=	Param	< or <=	Upper Bounds
0	unknown	2.41368	--		y0		--
1	center	30	--		xc__1		--
1	area	690.72758	--		A__1		--
1	Gaussian width	5.29802	0	<	wG__1	<=	5
1	Lorentzian width	5.29802	0	<	wL__1		--
2	center	70	--		xc__2		--
2	area	-309.8783	--		A__2		--
2	Gaussian width	5.29802	0	<	wG__2	<=	5
2	Lorentzian width	5.29802	0	<	wL__2		--
3	center	110	--		xc__3		--
3	area	-1276.0984	--		A__3		--
3	Gaussian width	5.29802	0	<	wG__3	<=	5
3	Lorentzian width	5.29802	0	<	wL__3		--
4	center	170	--		xc__4		--
4	area	481.05301	--		A__4		--
4	Gaussian width	5.29802	0	<	wG__4	<=	5
4	Lorentzian width	5.29802	0	<	wL__4		--

5. Click the **Fit Until Converged** button. When the fitting is done, click **OK** to close the dialog.
6. Back in the **Fit Peaks** page, click **Finish** to complete the analysis. See the results in the source workbook and the graph report.

9 Statistics

9.1 One Way ANOVA

9.1.1 Summary

There are two main modes of datasets in Statistics - indexed and raw. When you perform an analysis, you do not need to use the whole dataset, so Origin provides several ways to select data. For example, you can use the interactive Regional Data Selector button to graphically select the data or you can use the **Column Browser** dialog to make your selection.

In this tutorial, you'll use the Analysis of Variance (ANOVA) statistical test, to learn how to use these two different modes of data to perform analysis and how to select data by using the **Column Browser** dialog.

ANOVA is a kind of parametric method for means comparison and is an extension of t-test. When there are more than two groups to be compared, pairwise t-test is not appropriate and ANOVA should be used. ANOVA requires normality and equal variance. Otherwise, non-parametric analysis should be used.

Minimum Origin Version Required: Origin 8.0 SR6

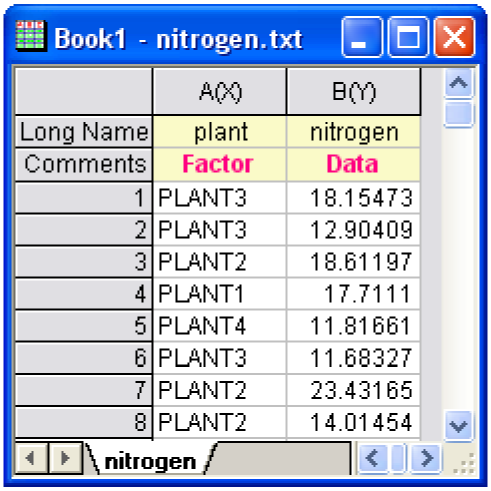
9.1.2 What you will learn

This tutorial will show you how to:

- Use different input data mode on statistical analysis dialog
- Test normality for special part of dataset
- Perform one-way ANOVA
- Select data by Column Browser

9.1.3 Steps

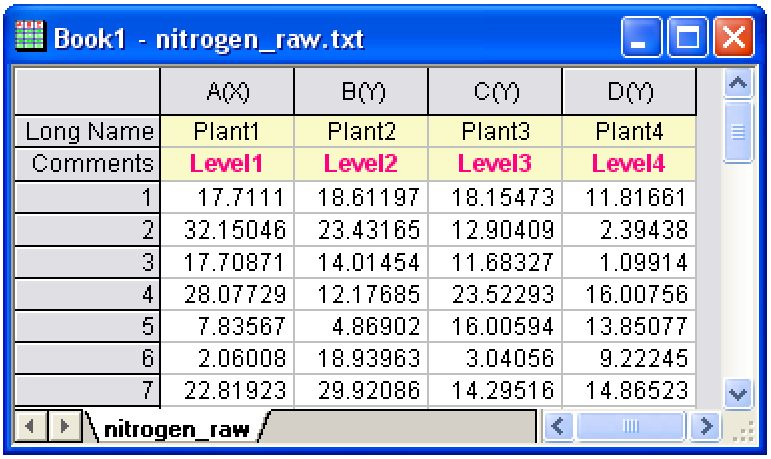
Origin can calculate ANOVA in indexed as well as raw data mode. For One-Way ANOVA, when using indexed mode, data should be organized in two columns : one for Factor and the other for data.



Book1 - nitrogen.txt

	A(X)	B(Y)
Long Name	plant	nitrogen
Comments	Factor	Data
1	PLANT3	18.15473
2	PLANT3	12.90409
3	PLANT2	18.61197
4	PLANT1	17.7111
5	PLANT4	11.81661
6	PLANT3	11.68327
7	PLANT2	23.43165
8	PLANT2	14.01454

When using Raw data mode, the different levels are in different columns.



Book1 - nitrogen_raw.txt

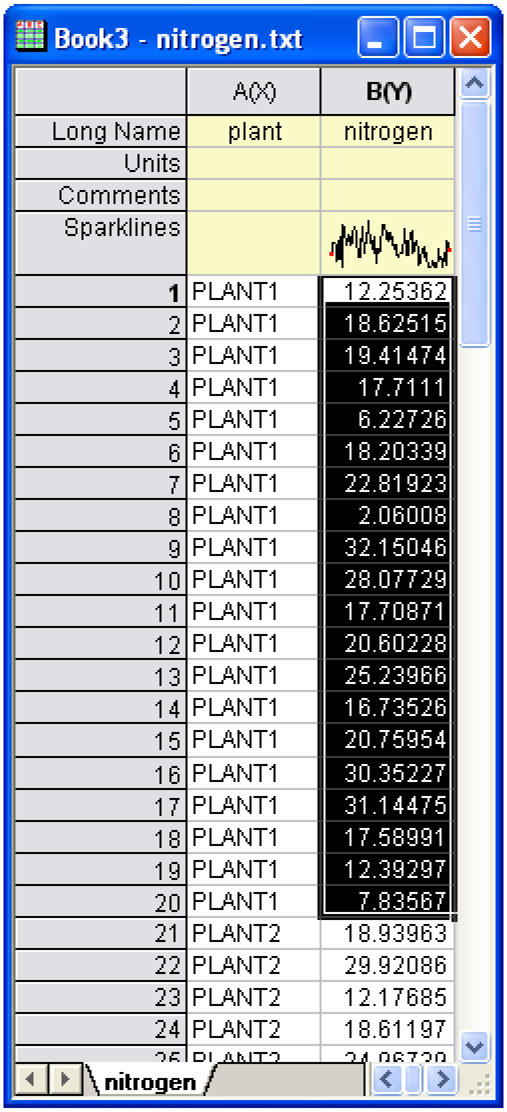
	A(X)	B(Y)	C(Y)	D(Y)
Long Name	Plant1	Plant2	Plant3	Plant4
Comments	Level1	Level2	Level3	Level4
1	17.7111	18.61197	18.15473	11.81661
2	32.15046	23.43165	12.90409	2.39438
3	17.70871	14.01454	11.68327	1.09914
4	28.07729	12.17685	23.52293	16.00756
5	7.83567	4.86902	16.00594	13.85077
6	2.06008	18.93963	3.04056	9.22245
7	22.81923	29.92086	14.29516	14.86523

Indexed data mode

Nitrogen content has been recorded in milligrams for 4 kinds of plant, and we are interested in whether different plants have different nitrogen content. We will perform One-Way ANOVA using index data mode for this example.

1. Start with a new workbook and import the file *\Samples\Statistics\nitrogen.txt*. Make sure you select *.txt* from the drop-down menu Files of type. First, we should perform a normality test on each group of data to determine if they are from a normal distribution.
2. Highlight the first column, right-click and select **Sort Worksheet** from the Worksheet menu and choose **Ascending**.

3. Highlight the second column from row 1 to row 20 - which belongs to "PLANT1" - and open the **Normality Test** dialog by choosing the menu item **Statistics: Descriptive Statistics: Normality Test**.



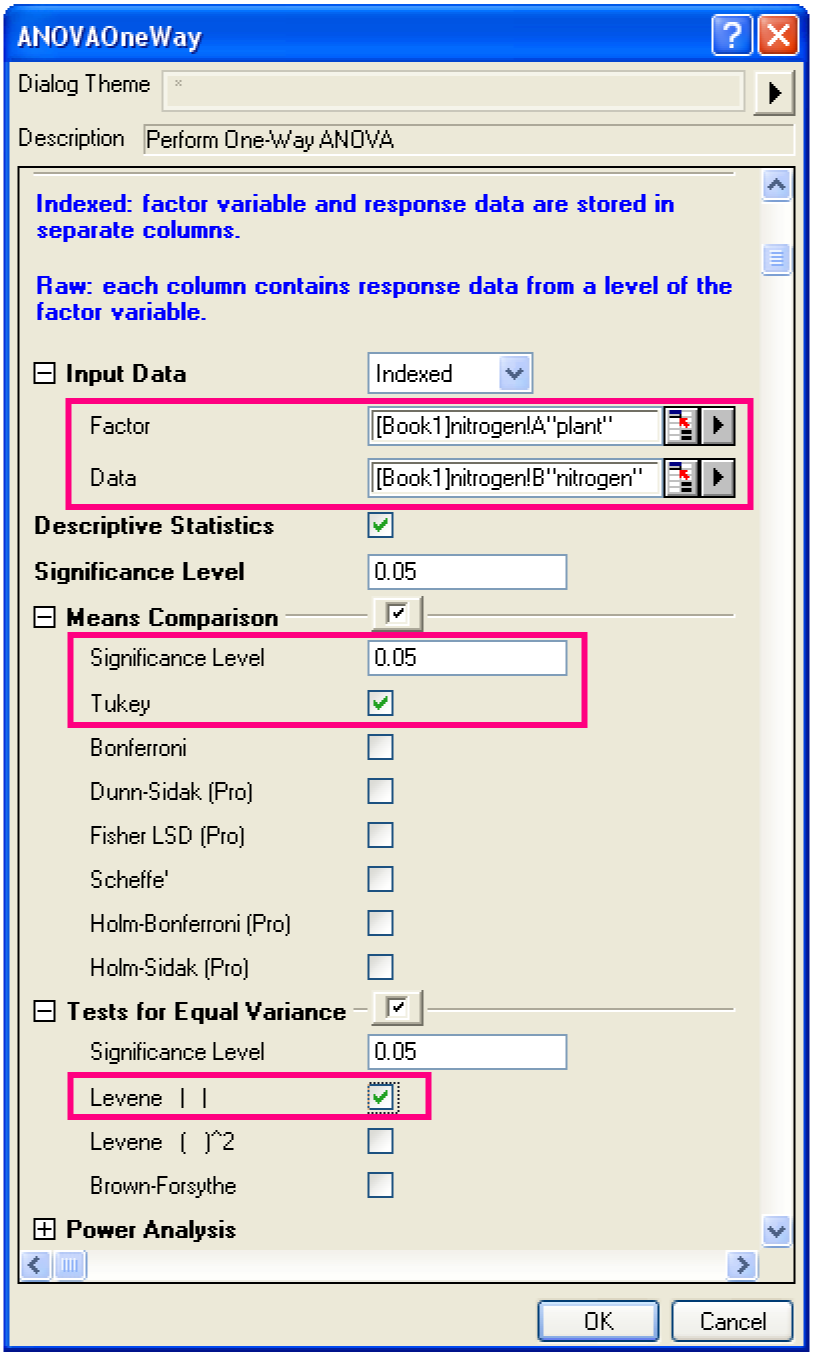
Book3 - nitrogen.txt

	A(X)	B(Y)
Long Name	plant	nitrogen
Units		
Comments		
Sparklines		
1	PLANT1	12.25362
2	PLANT1	18.62515
3	PLANT1	19.41474
4	PLANT1	17.7111
5	PLANT1	6.22726
6	PLANT1	18.20339
7	PLANT1	22.81923
8	PLANT1	2.06008
9	PLANT1	32.15046
10	PLANT1	28.07729
11	PLANT1	17.70871
12	PLANT1	20.60228
13	PLANT1	25.23966
14	PLANT1	16.73526
15	PLANT1	20.75954
16	PLANT1	30.35227
17	PLANT1	31.14475
18	PLANT1	17.58991
19	PLANT1	12.39297
20	PLANT1	7.83567
21	PLANT2	18.93963
22	PLANT2	29.92086
23	PLANT2	12.17685
24	PLANT2	18.61197
25	PLANT2	[illegible]

4. Use the default setting of the dialog and click **OK**. From the p-value of result, we can see "PLANT1" follows a normal distribution.

5. In a similar way, you can highlight the range of data "PLANT2", "PLANT3" and "PLANT4" and test for Normality. Our sample data has normal distribution for all plants.

6. With our nitrogen data worksheet active, open the **ANOVAOneWay** dialog by using the menu item **Statistics: ANOVA: One-Way ANOVA**. Set the **Input Data** mode as **Indexed**, assign the "plant" and "nitrogen" column as **Factor** and **Data** respectively using the right-arrow buttons. Click the + to expand the **Means Comparison** node, set <bSignificance Level</b> as 0.05 and check the **Tukey** Means Comparison method. Check **Levene | |** from **Tests for Equal Variance** branch. Click the **OK** button to perform One-Way ANOVA.



Dialog Theme *

Description Perform One-Way ANOVA

Indexed: factor variable and response data are stored in separate columns.

Raw: each column contains response data from a level of the factor variable.

⊟ Input Data Indexed

Factor [Book1]nitrogen!A"plant"

Data [Book1]nitrogen!B"nitrogen"

Descriptive Statistics ☑

Significance Level 0.05

⊟ Means Comparison ☑

Significance Level 0.05

Tukey ☑

Bonferroni □

Dunn-Sidak (Pro) □

Fisher LSD (Pro) □

Scheffe' □

Holm-Bonferroni (Pro) □

Holm-Sidak (Pro) □

⊟ Tests for Equal Variance ☑

Significance Level 0.05

Levene | | ☑

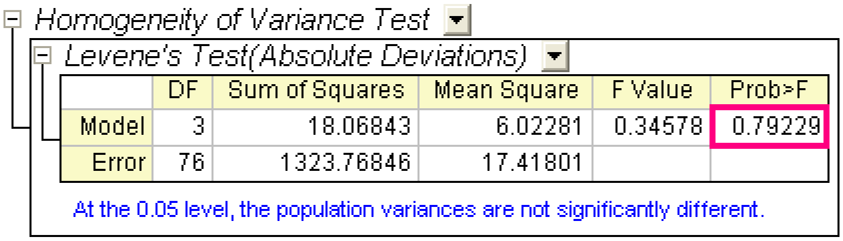
Levene ( )^2 □

Brown-Forsythe □

⊞ Power Analysis

Explaining the result:

- From the "Homogeneity of Variance Test" table of one-way ANOVA result, we can see that the four groups have equal variance, since the p-value is bigger than 0.05.



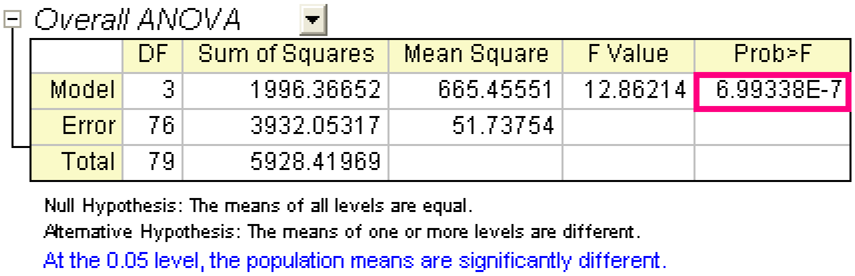
Homogeneity of Variance Test

Levene's Test(Absolute Deviations)

	DF	Sum of Squares	Mean Square	F Value	Prob>F
Model	3	18.06843	6.02281	0.34578	0.79229
Error	76	1323.76846	17.41801		

At the 0.05 level, the population variances are not significantly different.

- From the result of Overall ANOVA we can conclude that at least two groups of the four have significant different means, since the p-value is smaller than 0.05.

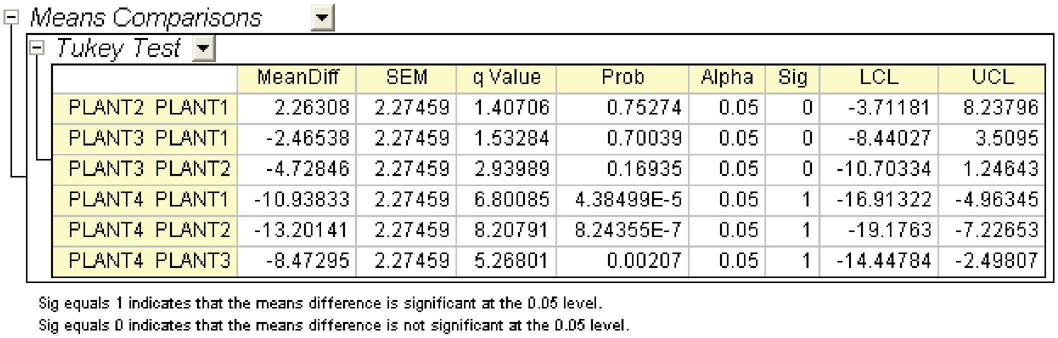


Overall ANOVA

	DF	Sum of Squares	Mean Square	F Value	Prob>F
Model	3	1996.36652	665.45551	12.86214	6.99338E-7
Error	76	3932.05317	51.73754		
Total	79	5928.41969			

Null Hypothesis: The means of all levels are equal.
Alternative Hypothesis: The means of one or more levels are different.
At the 0.05 level, the population means are significantly different.

- To research further, we expand the results of "Means Comparisons".



Means Comparisons

Tukey Test

	MeanDiff	SEM	q Value	Prob	Alpha	Sig	LCL	UCL
PLANT2 PLANT1	2.26308	2.27459	1.40706	0.75274	0.05	0	-3.71181	8.23796
PLANT3 PLANT1	-2.46538	2.27459	1.53284	0.70039	0.05	0	-8.44027	3.5095
PLANT3 PLANT2	-4.72846	2.27459	2.93989	0.16935	0.05	0	-10.70334	1.24643
PLANT4 PLANT1	-10.93833	2.27459	6.80085	4.38499E-5	0.05	1	-16.91322	-4.96345
PLANT4 PLANT2	-13.20141	2.27459	8.20791	8.24355E-7	0.05	1	-19.1763	-7.22653
PLANT4 PLANT3	-8.47295	2.27459	5.26801	0.00207	0.05	1	-14.44784	-2.49807

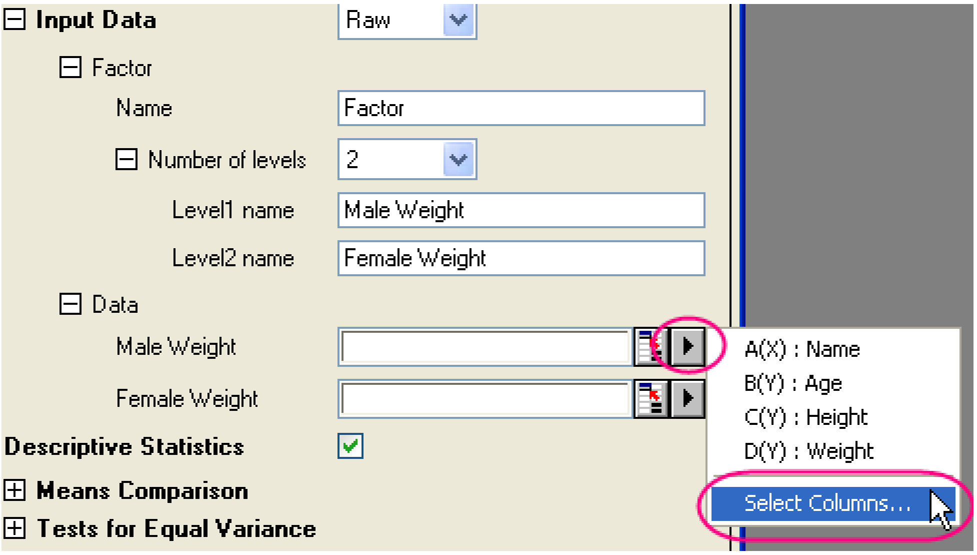
Sig equals 1 indicates that the means difference is significant at the 0.05 level.
Sig equals 0 indicates that the means difference is not significant at the 0.05 level.

Here we see that PLANT4 has significantly different means when compared to each of the other three groups.

raw data mode

1. Select **File : Open** and choose **WorkBooks** from **Files of type** drop-down list, and browse to *\Samples\Statistics* folder and open the file *Body.ogw*
2. Select menu item **Statistics : ANOVA : One-Way ANOVA** to bring up the **ANOVAOneWay** dialog. Choose **Raw** as **Input Data** mode. Enter the **Level1 Name** and **Level2 Name** as "Male Weight" and "Female Weight" respectively.

3. Now we will use the **Data Browser** to select data in the **Data** branch. Click the triangle icon beside **Male Weight** edit box, in the fly-out menu, select **Select Columns...** to open the **Column Browser** dialog.



⊟ **Input Data**: Raw

⊟ Factor

Name: Factor

⊟ Number of levels: 2

Level1 name: Male Weight

Level2 name: Female Weight

⊟ Data

Male Weight:

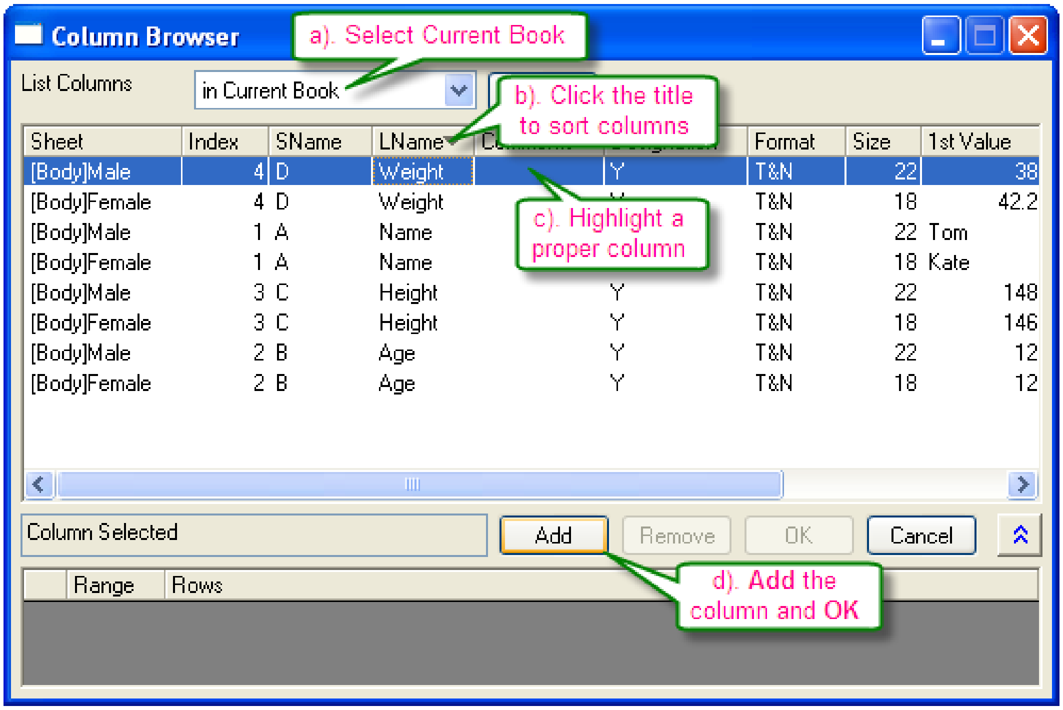
Female Weight:

Descriptive Statistics ☑

⊞ **Means Comparison**

⊞ **Tests for Equal Variance**

In the Column Browser dialog, you can select **in Current Book** from **List Columns** drop-down list to see all available worksheet columns in the current book. Select *Weight* in the sheet *[Body]Male* and click **Add** and **OK** to add it to **Male Weight** edit box. Similarly, assign *Weight* from *[Body]Female* to **Female Weight** edit box.



Column Browser

List Columns: in Current Book

Sheet	Index	SName	LName	C		Format	Size	1st Value
[Body]Male	4	D	Weight		Y	T&N	22	38
[Body]Female	4	D	Weight		Y	T&N	18	42.2
[Body]Male	1	A	Name			T&N	22	Tom
[Body]Female	1	A	Name			T&N	18	Kate
[Body]Male	3	C	Height		Y	T&N	22	148
[Body]Female	3	C	Height		Y	T&N	18	146
[Body]Male	2	B	Age		Y	T&N	22	12
[Body]Female	2	B	Age		Y	T&N	18	12

Column Selected

	Range	Rows

Add
Remove
OK
Cancel
a). Select Current Book
b). Click the title to sort columns
c). Highlight a proper column
d). Add the column and OK

4. Accept other default settings in the ANOVAOneWay dialog and click **OK**. From the output report footnote, we can conclude that at the 0.05 level, the population weight means between male and female are not significantly different.

10 Automation

- Creating and Using Analysis Templates
- Creating a Custom Report Sheet
- Batch Processing using Import Wizard and Analysis Template
- Creating Analysis Template using Set Column Value

10.1 Creating and Using Analysis Templates

10.1.1 Summary

Routine tasks can be simplified by creating an Analysis Template. Such templates can contain multiple analysis results and also custom report sheets. A new instance of the template can then be opened any time and source data can be changed to update all analysis results and custom reports.

Minimum Origin Version Required: Origin 8.0 SR6

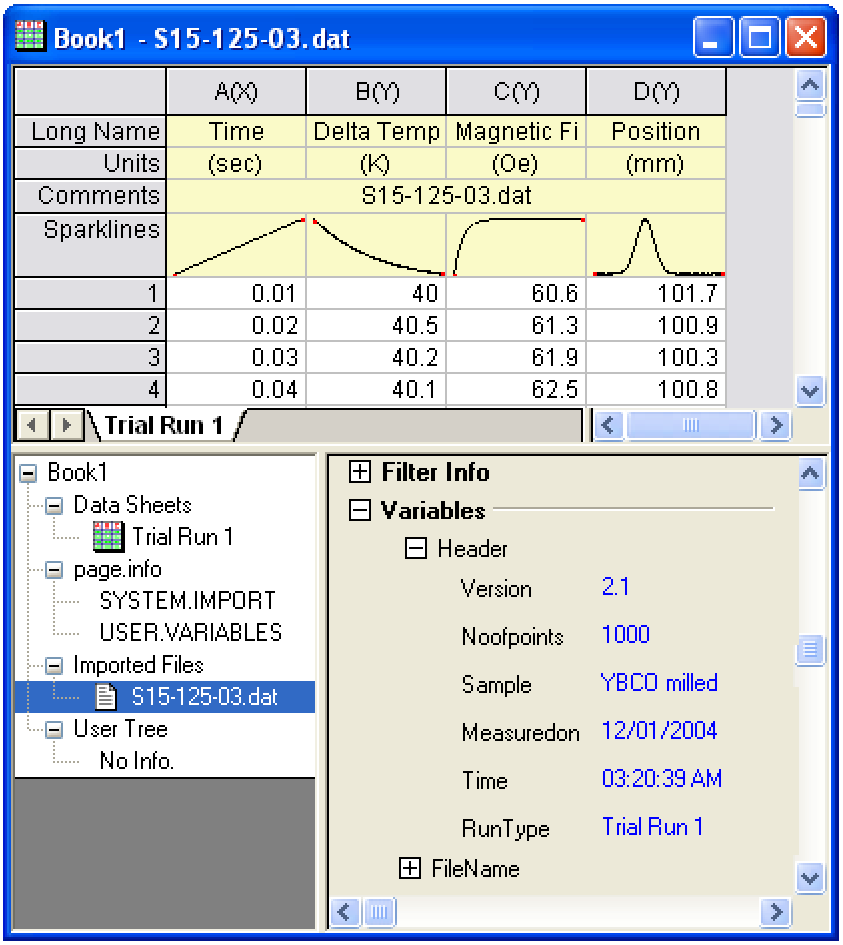
10.1.2 What you will learn

- How to create analysis template (OGW)
- How to re-use analysis template with new data

10.1.3 Steps

Importing Data

1. Start with a new workbook.
2. Click the **Import Wizard** button on the Standard Toolbar, or invoke the **Import: Import Wizard...** menu item. the Import Wizard dialog will open.
3. Click the **...** button next to the **File** edit box and select the file **<Origin EXE folder>\Samples\Import and Export\S15-125-03.dat**.
4. Verify that the **Import Filters for Current Data Type** drop-down shows **Data Folder: VarFromFileNameAndHeader**.
5. Change the **Import Mode** drop-down to **Replace Existing Data**.
6. This is an import filter shipped with the sample file, that specifies how to import the file and what header and file name strings to parse to create import variables. Walk through the wizard pages to view the settings (Optional) and then click Finish button to import the file.
7. Right-click on workbook title bar and select **Show Organizer** to turn on organizer panel. Expand branches and verify that variables have been created and saved, as in the picture below:



Book1 - S15-125-03.dat

	A(X)	B(Y)	C(Y)	D(Y)
Long Name	Time	Delta Temp	Magnetic Fi	Position
Units	(sec)	(K)	(Oe)	(mm)
Comments	S15-125-03.dat			
Sparklines				
1	0.01	40	60.6	101.7
2	0.02	40.5	61.3	100.9
3	0.03	40.2	61.9	100.3
4	0.04	40.1	62.5	100.8

Version	2.1
Noofpoints	1000
Sample	YBCO milled
Measuredon	12/01/2004
Time	03:20:39 AM
RunType	Trial Run 1

Performing Analysis

1. Highlight column D and use the **Analysis: Fitting: Nonlinear Curve Fit...** menu item to open the **NLFit** dialog.
2. Fit the data with **Gauss** function. This will add a hierarchical report sheet to the book, with result tables and embedded graph with data and fit curve.
3. Go to the **FitNL1** report sheet and double-click to open the graph containing data and fit curve. Perform some customization of the graph such as adding grid lines, changing font size etc. Click the **X** button on the graph window to put the modified graph back into the report.

Saving the Analysis Template

1. Go to the source data sheet of the workbook, which should be the first sheet. Select the **Worksheet: Clear Worksheet...** menu item and press **OK** in the dialog that opens. This will clear all the data from the sheet. The analysis report sheet will now be empty. Clearing the data is optional, and it makes the size of the analysis template file to be smaller.
2. Use the **File: Save Window As...** menu item and save the book as an **OGW** file under your **User Files Folder** with a suitable name such as **Analysis Template**. This OGW file can now serve as an Analysis Template for future analysis of similar data.

Re-using the Analysis Template

1. Start a new project and then select the menu item **File: Recent Books** and from the fly-out options select the Analysis Template saved earlier.
2. Make the data sheet active, and select **File: Import Wizard...** and select the file **<Origin EXE path>\Samples\Import and Export\S21-235-07.dat**.
3. Make sure the filter drop-down shows **VarsFromFileNameAndHeader** and change the **Import Mode** drop-down to **Replace Existing Data** and click **Finish**.
4. Press the **Recalculate** button, which is the last button on the **Standard** toolbar. Origin will recalculate the analysis results and update the custom report sheet links, and at this point you can view and print the custom report sheet.

10.2 Creating a Custom Report Sheet

10.2.1 Summary

Worksheets in Origin can be customized by merging cells and placing various objects such as graphs, external images, links to variables and tables/cells in other sheets, in order to create custom reports. Such custom reports can be part of an analysis template, thus allowing user to open the analysis template, change data, and simply print their updated custom report.

Minimum Origin Version Required: Origin 8.0 SR6

10.2.2 What you will learn

- How to create a custom report sheet
- How to save custom report as part of Analysis Template (OGW) and re-use with new data

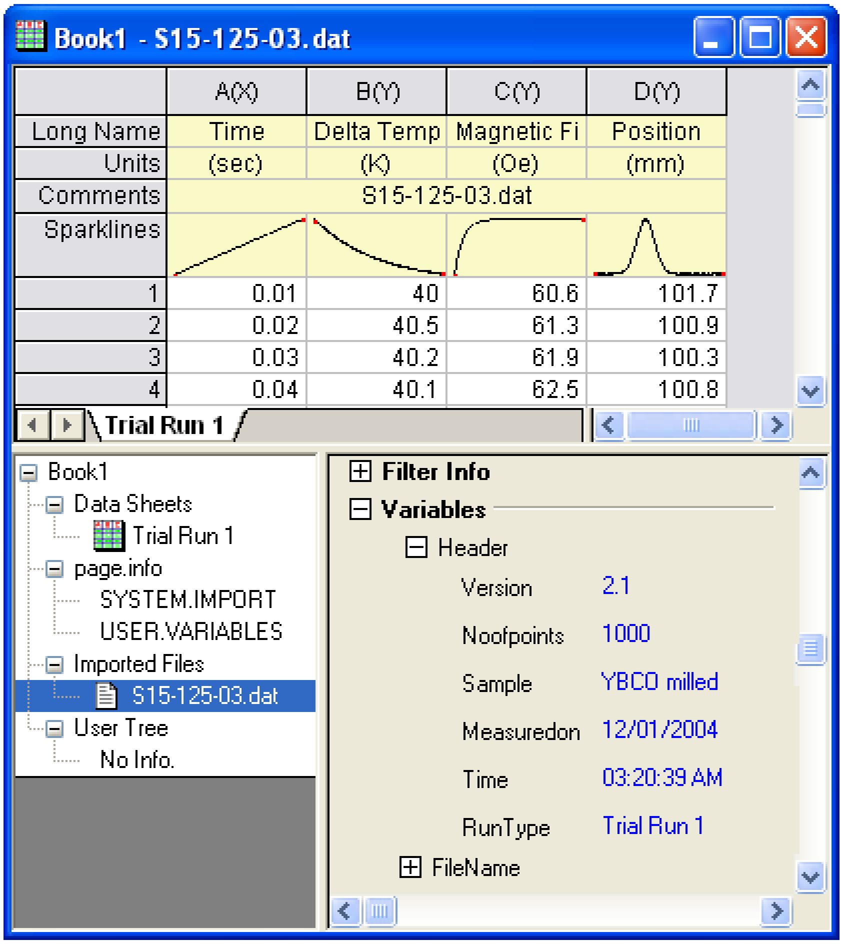
10.2.3 Steps

Note: First finish the previous tutorial named "Creating and Using Analysis Templates" where an analysis template named **Analysis Template.OGW** is created.

Importing Data

1. Use the **File:Open** menu item and open the Analysis Template **My Custom Analysis.OGW**. This analysis template already has a nonlinear fitting analysis operation set up for data from column D of the first sheet.
2. Click the **Import Wizard** button on the Standard Toolbar, or invoke the **Import: Import Wizard...** menu item. the Import Wizard dialog will open.
3. Click the **...** button next to the **File** edit box and select the file **<Origin EXE folder>\Samples\Import and Export\S15-125-03.dat**.
4. Verify that the **Import Filters for Current Data Type** drop-down shows "Data Folder: VarFromFileNameAndHeader".
5. Verify that the **Import Mode** drop-down is set to **Replace Existing Data**.
6. This is an import filter shipped with the sample file, that specifies how to import the file and what header and file name strings to parse to create import variables. Walk thru the wizard pages to view the settings (Optional) and then click Finish button to import the file.

7. Right-click on workbook title bar and select **Show Organizer** to turn on organizer panel. Expand branches and verify that variables have been created and saved, as in the picture below:



Book1 - S15-125-03.dat

	A(X)	B(Y)	C(Y)	D(Y)
Long Name	Time	Delta Temp	Magnetic Fi	Position
Units	(sec)	(K)	(Oe)	(mm)
Comments	S15-125-03.dat			
Sparklines				
1	0.01	40	60.6	101.7
2	0.02	40.5	61.3	100.9
3	0.03	40.2	61.9	100.3
4	0.04	40.1	62.5	100.8

Trial Run 1

- Book1
  - Data Sheets
    - Trial Run 1
  - page.info
    - SYSTEM.IMPORT
    - USER.VARIABLES
  - Imported Files
    - S15-125-03.dat
  - User Tree
    - No Info.

⊞ Filter Info

⊟ Variables

⊟ Header

Version	2.1
Noofpoints	1000
Sample	YBCO milled
Measuredon	12/01/2004
Time	03:20:39 AM
RunType	Trial Run 1

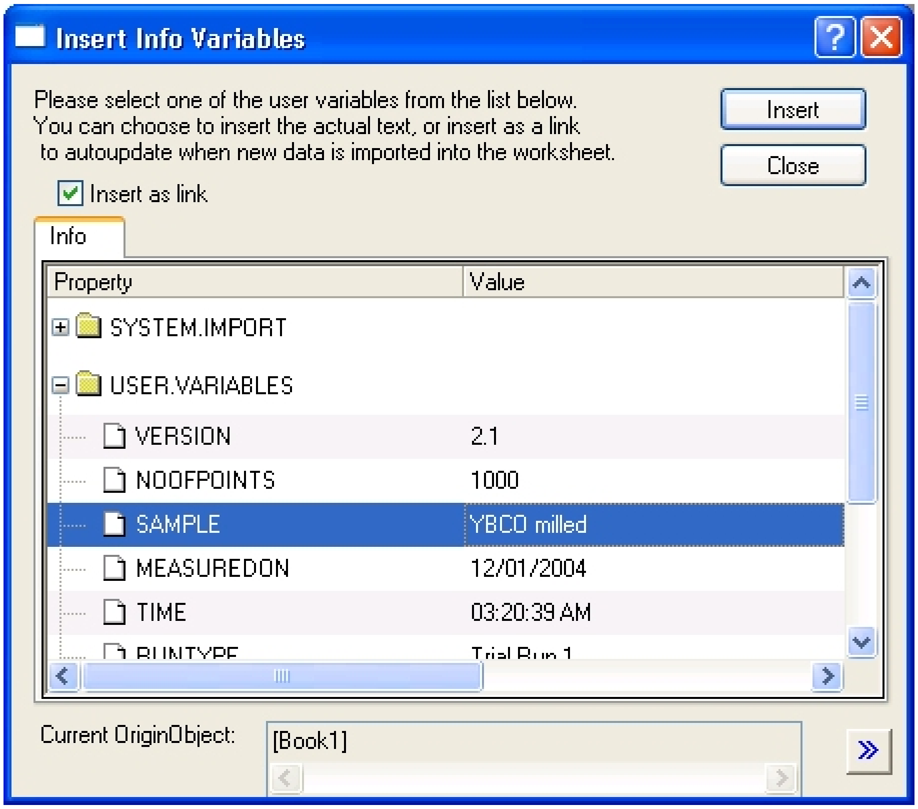
⊞ FileName

8. Press the **Recalculate** button on the **Standard** toolbar to update the analysis result sheet. Verify that the analysis was updated and the embedded graph shows the new raw data and fit curve.

Creating Custom Report Sheet

1. Right click on one of the worksheet tabs and select **Add** to add a new worksheet. Rename this worksheet as **Custom Report**.
2. Make the **Custom Report** sheet active and add multiple blank columns.
3. Go to **FitNL1** worksheet and right click on the graph with data and fit curve and select **Copy** from the context menu.
4. Go to **Custom Report** and right click in 1st column in a middle row and select **Paste Link**. This will paste a link to the embedded graph. Click and select a group of cells with this pasted cell at the top-left. Then click the **Merge Cells** button, which is the last button on the **Styles** toolbar. This will merge the group of cells and the graph image will be shown larger in size.
5. Go to **FitNL1** report sheet and right click on the **Parameters** node and select **Copy Table** from the context menu.

6. Go to the **Custom Report** sheet and right click in a cell to the right of the graph and select **Paste Link** from the context menu. This will place links to all values of the parameter table entries in the custom report.
7. Select the numeric value cells and right-click and select **Format Cells...** to bring up format dialog. Change the **Digits** drop-down to **Set Decimal Places=** and enter **2** in the **Decimal Number** edit box and click **OK** to format the numbers.
8. Select various cells in the table and use the **Style** toolbar controls to change foreground and background color, and use the **Standard** toolbar to change font size etc.
9. Right-click on top-left cell in the custom report sheet and select **Insert Images from Files...** context menu and select some image such as a company logo image. Click and drag to cover more cells and then click the **Merge Cells** button to increase the size of the logo display.
10. Click inside a cell on top-right and type in the string **var://@D** and press **Enter**. Right click on cell and select **Format Cells...**, set the **Format** as **Date**, and then elect a suitable format from the **Display** drop-down. This will place the current date, pointed to by @D LabTalk variable, into the worksheet cell. Click and expand the selection to multiple cells and press the **Merge Cells** button to show the date with larger font size.
11. Right-click on a cell below the logo and date, and select **Insert Variables** context menu. In the dialog that opens, select **User.Variables** branch and select **Sample**. Check the **Insert as Link** check box on top and press **Insert** to insert variable as link into the report sheet.



Please select one of the user variables from the list below.
You can choose to insert the actual text, or insert as a link
to autoupdate when new data is imported into the worksheet.

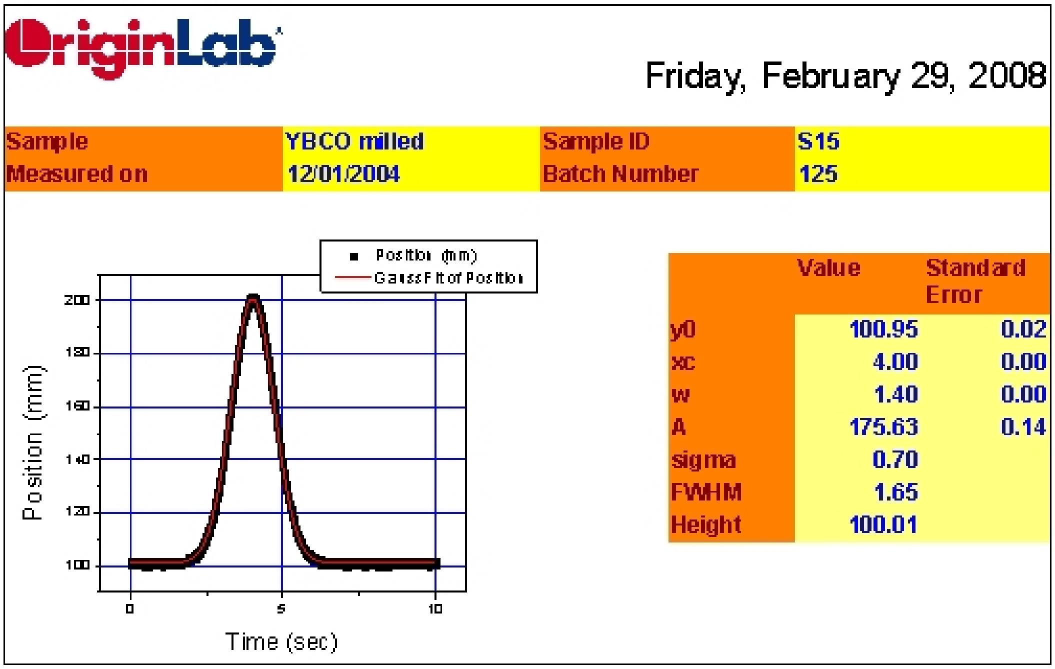
☑ Insert as link

Property	Value
⊞ SYSTEM.IMPORT	
⊟ USER.VARIABLES	
VERSION	2.1
NOOFPOINTS	1000
SAMPLE	YBCO milled
MEASUREDON	12/01/2004
TIME	03:20:39 AM
RUNTYPE	Trial Run 1

Current OriginObject: [Book1]

Click on a cell to the left of the inserted variable, and enter the static text **Sample**.

12. Insert more variables and format the cells for color and font.
13. Invoke the **Format: Worksheet...** menu item to open the **Worksheet Properties** dialog. Under the **View** tab, expand **Show Headers** and uncheck the column and row header check boxes. Expand the **show Grid Lines** branch and uncheck the column and row grid check boxes. Select the **Format** tab and check the **Show Missing as Blank** check box. Click **OK** to close this dialog.
14. Right-click on worksheet title bar and select **View: Long Name** to turn off long name row. Also turn off **Units** and **Comments**.
15. Select the **File: Print** menu item to open the print dialog, and press **Options** button, and uncheck the Horizontal/Vertical grid lines. Select **File: Print Preview**. Your custom report sheet should look like the image below:



Friday, February 29, 2008

Sample	YBCO milled	Sample ID	S15
Measured on	12/01/2004	Batch Number	125

	Value	Standard Error
y0	100.95	0.02
xc	4.00	0.00
w	1.40	0.00
A	175.63	0.14
sigma	0.70	
FWHM	1.65	
Height	100.01	

Saving the Analysis Template

1. Go to the source data sheet of the workbook, which should be the first sheet. Select the **Worksheet: Clear Worksheet...** menu item and press **OK** in the dialog that opens. This will clear all the data from the sheet. The analysis report sheet and the custom report sheet will now be empty. Clearing the data is optional, and it makes the size of the analysis template file to be smaller.
2. Use the **File: Save Window As...** menu item and save the book as an **OGW** file under your **User Files Folder** with a suitable name such as **My Custom Analysis**. This OGW file can now serve as an Analysis Template for future analysis of similar data.

Re-using the Analysis Template

1. Start a new project and then select the menu item **File: Recent Books** and from the fly-out options select the Analysis Template saved earlier.

2. Make the data sheet active, and select **File: Import Wizard...** and select the file **<Origin EXE path>\Samples\Import and Export\S21-235-07.dat**.
3. Make sure the filter drop-down shows **VarsFromFileNameAndHeader** and change the **Import Mode** drop-down to **Replace Existing Data** and click **Finish**.
4. Press the **Recalculate** button, which is the last button on the **Standard** toolbar. Origin will recalculate the analysis results and update the custom report sheet links, and at this point you can view and print the custom report sheet.

10.3 Batch Processing using Import Wizard and Analysis Template

10.3.1 Summary

Once an Analysis Template (OGW) file has been created that optionally contains a custom report, the template could then be used to perform batch processing of multiple files, from the GUI as well as from script. This example shows how to perform batch processing using the Import Wizard and an Analysis Template.

Minimum Origin Version Required: Origin 8.0 SR6

10.3.2 What you will learn

- How to use Import Wizard and an Analysis Template for batch processing

10.3.3 Steps

Note: First finish the previous tutorial named "Creating a Custom Report Sheet" where an analysis template named **Analysis Template with Custom Report.OGW** is created.

Batch Processing with Import Wizard

1. Use the **File:Open** menu and open the Analysis Template named **Analysis Template with Custom Report.OGW**
2. Use the **Save Window As...** menu item and save this template (with same name, or another name) to your User Files (UFF) folder. Only analysis templates saved in UFF and Group folder are accessible from the Import Wizard.
3. Start a new project and open the **Import Wizard** dialog.
4. Select all three files: **<Origin EXE path>\Samples\Import and Export\S15-125-03.dat, S21-235-07.dat, S32-014-04.dat**.
5. Verify that the filter drop-down shows **VarsFromFileNameAndHeader**.
6. Click on the **Template** drop-down and select the Analysis Template that you saved earlier to the UFF area.
7. Change the **Import Mode** drop-down to **Start New Books** and click Finish.
8. Each file will be imported into a new book. Now press the **Recalculate** button on the **Standard** toolbar. All operations will be updated and each book will have an updated report sheet that you can view and print.

10.4 Creating Analysis Templates using Set Column Value

10.4.1 Summary

This tutorial will demonstrate how to add a column, set up **Before Formula Script** and have that script run whenever data changes in other columns. This technique can be used to create an **Analysis Template** for repeated analysis of similar data.

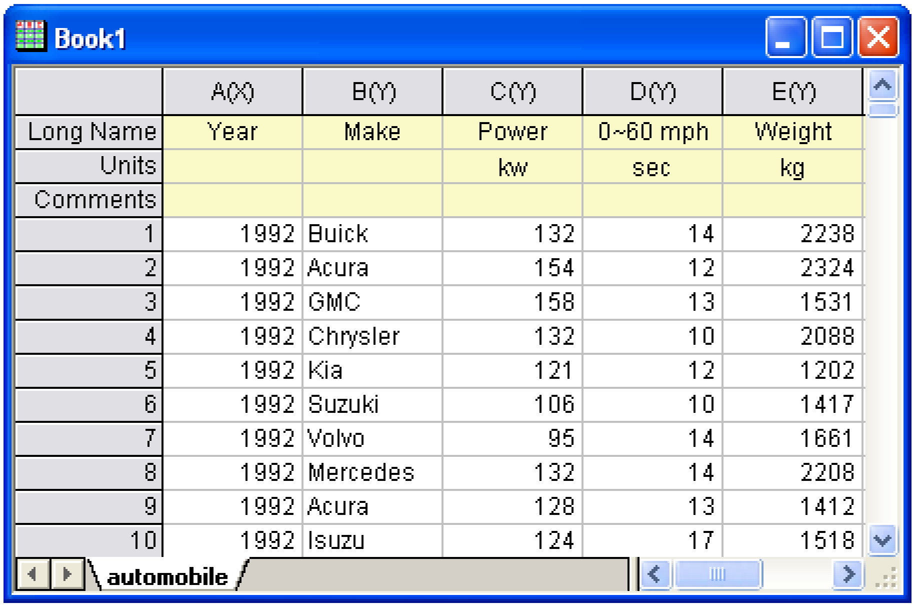
Minimum Origin Version Required: Origin 8.0 SR6

10.4.2 What you will learn

- How to use Set Column Values to create an analysis template
- How to select rows via the *Go to* function

10.4.3 Steps

1. Import the data from *\Samples\Statistics\automobile.dat* into a newly created workbook, as below. In this example, we will extract data, according to the *Make* column, into different worksheets.



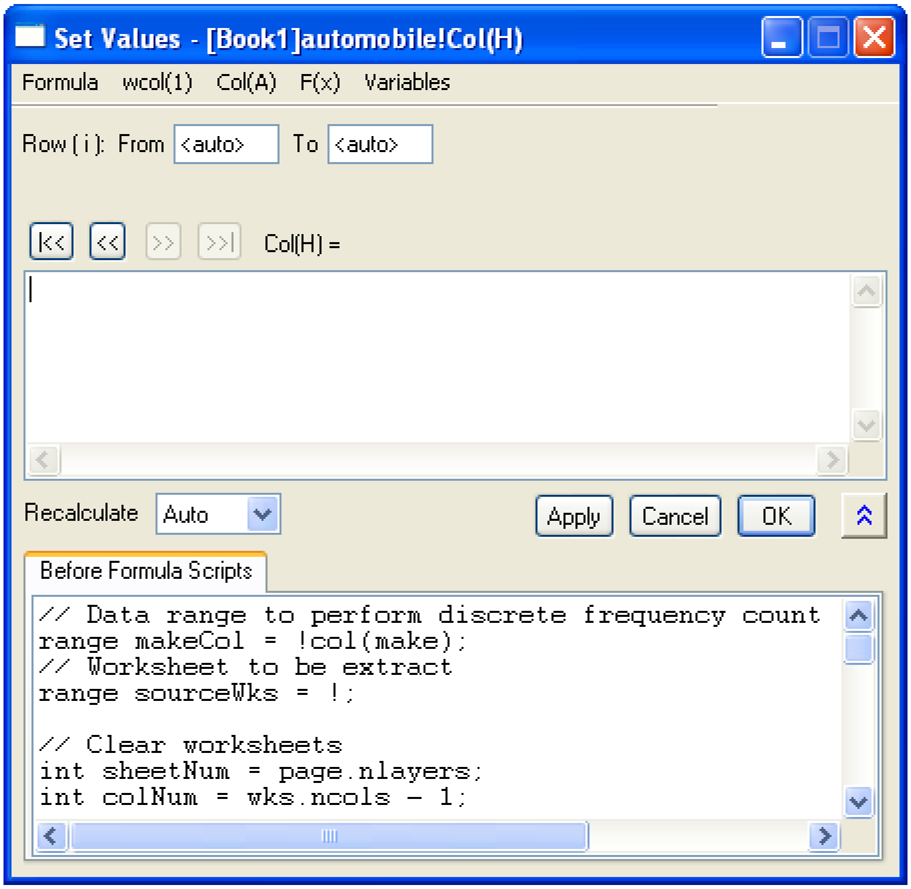
	A(X)	B(Y)	C(Y)	D(Y)	E(Y)
Long Name	Year	Make	Power	0~60 mph	Weight
Units			kw	sec	kg
Comments					
1	1992	Buick	132	14	2238
2	1992	Acura	154	12	2324
3	1992	GMC	158	13	1531
4	1992	Chrysler	132	10	2088
5	1992	Kia	121	12	1202
6	1992	Suzuki	106	10	1417
7	1992	Volvo	95	14	1661
8	1992	Mercedes	132	14	2208
9	1992	Acura	128	13	1412
10	1992	Isuzu	124	17	1518

2. Add an empty column to the worksheet and bring up the **Set Column Values** dialog of the column. In the **Before Formula Script** box, enter the script below.

```
// Data range on which to perform discrete frequency count
range makeCol = !col(make);
// Worksheet to be extracted
range sourceWks = !;

// Clear worksheets
int sheetNum = page.nlayers;
int colNum = wks.ncols - 1;
```

```
if (sheetNum>1)
{
        for (jj=2; jj<=sheetNum; jj++)
        {
                layer -d 2;
        }
}

// Tree variable to hold discfreqs outputs
tree tr;
// Perform discrete frequency count
discfreqs irng:=makeCol rd:=tr;
// String array to get result from tree
StringArray sa;
sa.append(tr.FreqCount1.Data1);

if( sa.GetSize() != NANUM )
{
        // Loop to extract data
        for (ii=1; ii<=sa.GetSize(); ii++)
        {
                string sn$ = sa.GetAt(ii)$;
                // Extract condition string
                string cond$ = "makeCol$ = " + sn$;
                // Create worksheet with different Make name
                newsheet name:=sn$ cols:=colNum outname:=on$ active:=0;
                // Extract data
                wxt test:=cond$ iw:=sourceWks c2:=colNum ow:=on$;
        }
}
```

3. This script will first perform a discrete frequency count on the *Make* column to get distinct values for *Make*. It will then create a new worksheet for each brand and extract data into these sheets.

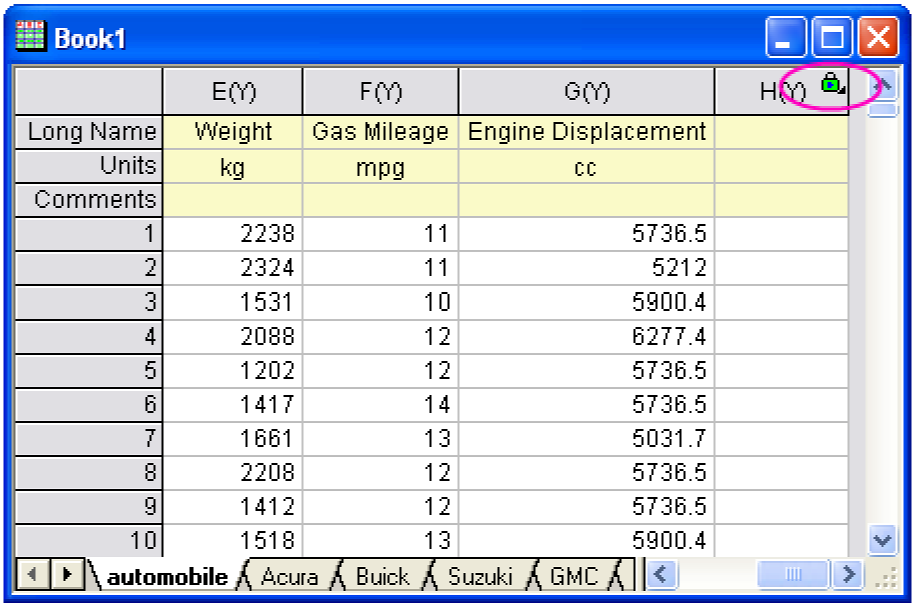


Set Values - [Book1]automobile!Col(H)
Formula wcol(1) Col(A) F(x) Variables
Row (i): From <auto> To <auto>
Col(H) =
Recalculate Auto
Apply Cancel OK
Before Formula Scripts

```
// Data range to perform discrete frequency count
range makeCol = !col(make);
// Worksheet to be extract
range sourceWks = !;

// Clear worksheets
int sheetNum = page.nlayers;
int colNum = wks.ncols - 1;
```

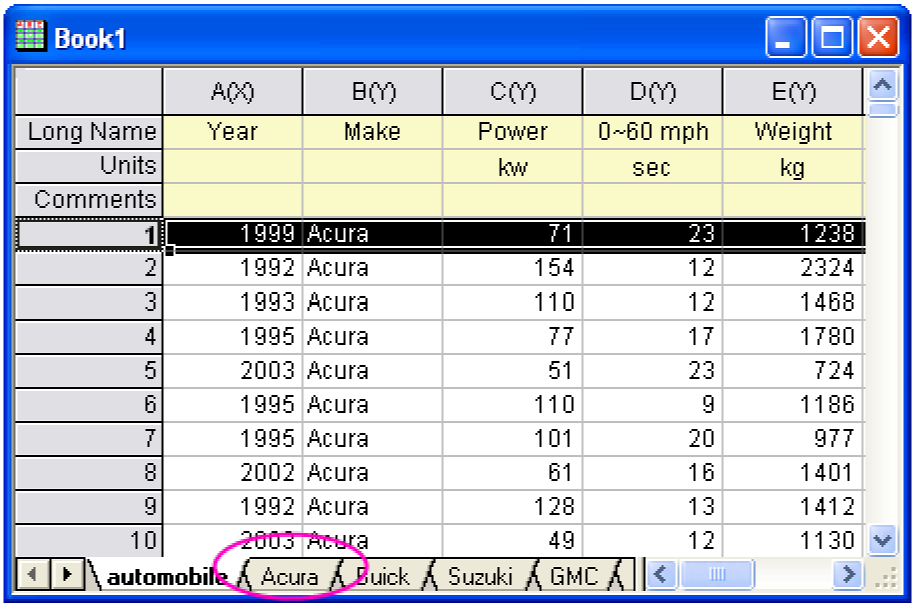
4.

5. Make sure the recalculate mode is set to **Auto** and click **OK**. The data will be separated into different worksheets. Then the empty column (H) with a green lock icon indicates that this procedure can be updated automatically.



	E(Y)	F(Y)	G(Y)	H(Y)
Long Name	Weight	Gas Mileage	Engine Displacement	
Units	kg	mpg	cc	
Comments				
1	2238	11	5736.5	
2	2324	11	5212	
3	1531	10	5900.4	
4	2088	12	6277.4	
5	1202	12	5736.5	
6	1417	14	5736.5	
7	1661	13	5031.7	
8	2208	12	5736.5	
9	1412	12	5736.5	
10	1518	13	5900.4	

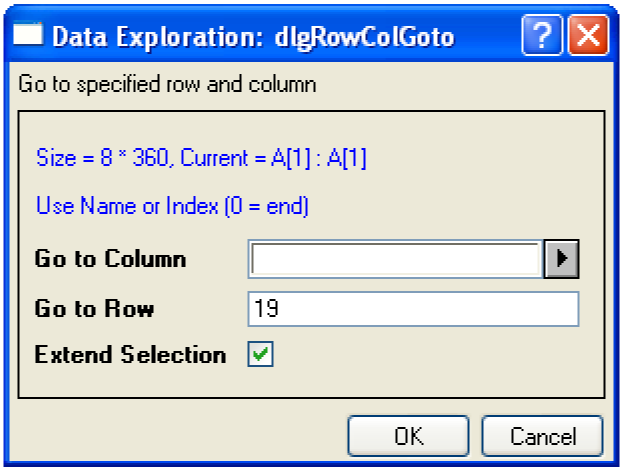
6. There are 18 makes of cars in the source data, so 18 new worksheets were created. Now we can check whether auto-update works.
   Highlight the *Make* column and right-click on it. Sort the worksheet ascending by this column (Actually, the auto-update may already be triggered after sorting). Now we can see there are 19 rows for Acura, and the worksheet Acura is just a nearby raw data sheet.



Book1

	A(X)	B(Y)	C(Y)	D(Y)	E(Y)
Long Name	Year	Make	Power	0~60 mph	Weight
Units			kw	sec	kg
Comments					
1	1999	Acura	71	23	1238
2	1992	Acura	154	12	2324
3	1993	Acura	110	12	1468
4	1995	Acura	77	17	1780
5	2003	Acura	51	23	724
6	1995	Acura	110	9	1186
7	1995	Acura	101	20	977
8	2002	Acura	61	16	1401
9	1992	Acura	128	13	1412
10	2003	Acura	49	12	1130

7. Highlight all of row 1, and select **Edit: Go to**. Enter 19 in the **Go to Row** edit box, and check the **Extend Selection** check box.



Data Exploration: dlgRowColGoto

Go to specified row and column

Size = 8 * 360, Current = A[1] : A[1]

Use Name or Index (0 = end)

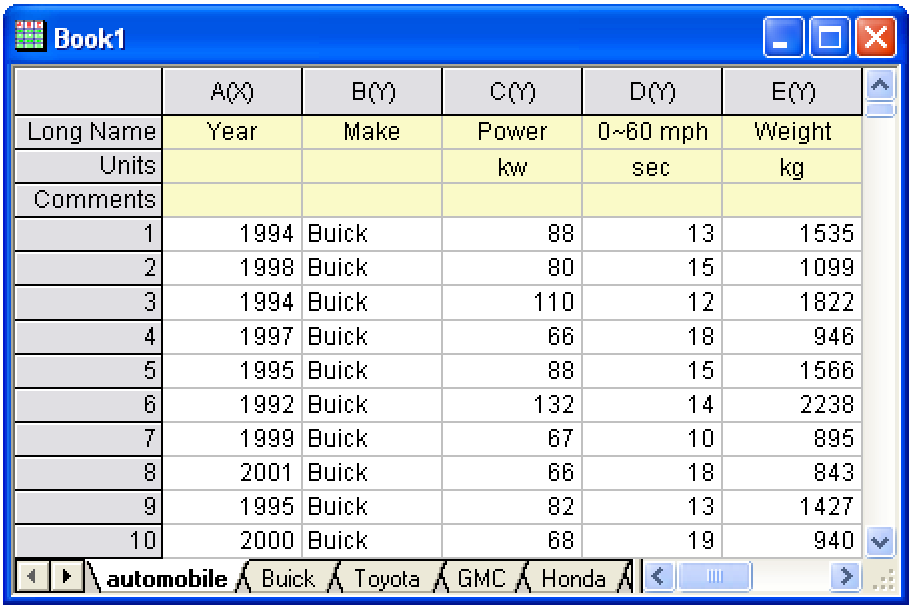
Go to Column

Go to Row 19

Extend Selection ☑

Click **OK**; now all the Acura rows are selected.

8. Right-click on the selection and choose **Delete**. Then auto-update will be triggered and there will no longer be any Acura output.



	A(X)	B(Y)	C(Y)	D(Y)	E(Y)
Long Name	Year	Make	Power	0~60 mph	Weight
Units			kw	sec	kg
Comments					
1	1994	Buick	88	13	1535
2	1998	Buick	80	15	1099
3	1994	Buick	110	12	1822
4	1997	Buick	66	18	946
5	1995	Buick	88	15	1566
6	1992	Buick	132	14	2238
7	1999	Buick	67	10	895
8	2001	Buick	66	18	843
9	1995	Buick	82	13	1427
10	2000	Buick	68	19	940

11 Programming

Origin's publication-quality technical graphics are supported by an equally impressive array of data import and analysis features. We have designed these tutorials to help you get a feel for the breadth of Origin's capabilities.

11.1 Command Window and X-Functions

11.1.1 Summary

Many of Origin's analysis tools and other data processing tools have been implemented using X-Functions. The Command Window provides a convenient way to run these functions.

Another important use for the Command Window is to send LabTalk script commands to Origin. Script commands can range from simple math and data operations, to user-created X-Functions or Origin C functions.

Minimum Origin Version Required: Origin 8.0 SR6

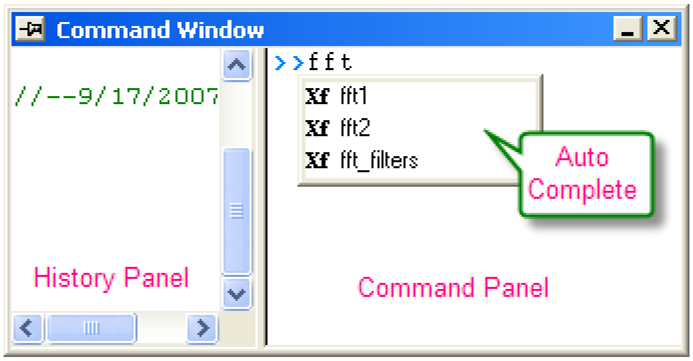
11.1.2 What you will learn

This tutorial will introduce you to the Command Window and show you how to:

- Perform simple calculation
- Access worksheet cells/columns
- Access X-Functions

11.1.3 Command Window

The Command Window consists of two panels: the **Command Panel** and **History Panel**:



```
//--9/17/2007
```

```
>>fft
Xf fft1
Xf fft2
Xf fft_filters
```

The Command Window is normally located at the bottom right corner of the screen, but if it is not visible, you can access it by pressing **Alt+3** or by selecting **View: Command Window**.

When typing in the Command Panel, the Auto Complete support allows you to choose among X-Function script commands and OGS files in the current working folder. The command and OGS file name will be respectively preceded by Xf and LT. You can move up and down the list using the arrow keys; pressing Enter selects the item. After your selection, press the space bar and the Auto Complete now shows you the available options for the command.

11.1.4 Examples

Perform Calculations

The Command Window can be used as a calculator or to access any of Origin's mathematical functions. See the examples below.

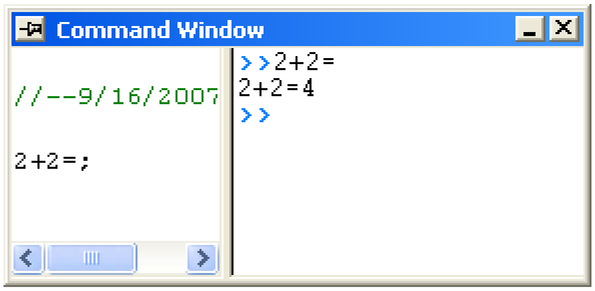
Single Line

One of the more basic uses for the Command Window is as an interface to perform simple calculations. For example, type the following:

```
2+2=
```

Press **ENTER**. Origin returns

```
2+2=4
```



```
//--9/16/2007

2+2=;
```

```
>>2+2=
2+2=4
>>
```

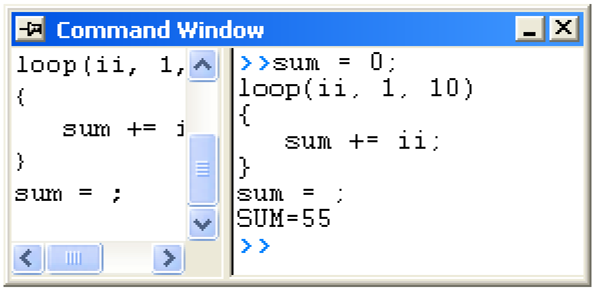
Multiple Lines

If you are typing multiple lines of scripts, first edit it in Code Builder (**View:Code Builder**) or any text editor, such as Windows Notepad, ending each line with a semi-colon, and then Copy + Paste the script in the Command Window, and press **ENTER** to execute. For example, paste the following script in the Command Window and **ENTER**:

```
sum = 0;
loop(ii, 1, 10)
{
   sum += ii;
}
sum = ;
```

Origin returns:

```
SUM=55
```



```
loop(ii, 1,
{
    sum += i
}
sum = ;
```

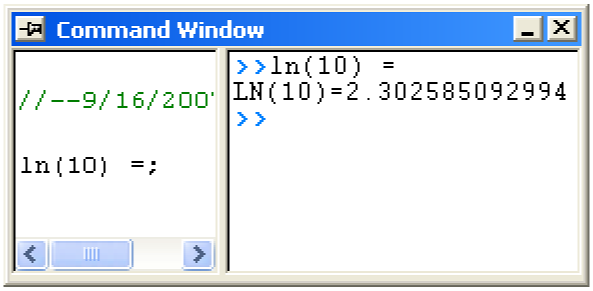
```
>>sum = 0;
loop(ii, 1, 10)
{
    sum += ii;
}
sum = ;
SUM=55
>>
```

Functions

Any mathematical function, built-in as well as user-created, can be executed from the Command Window. For example, type:

```
ln(10) =
```

Origin returns natural logarithm value of 10.



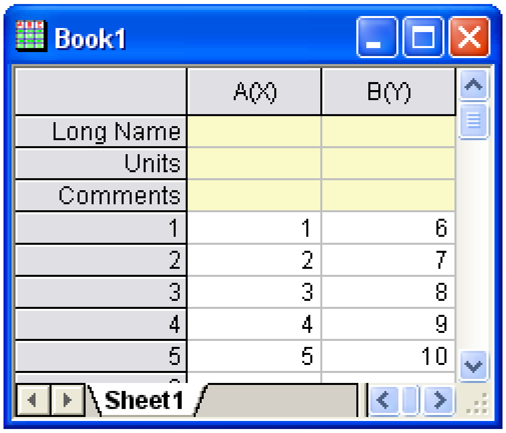
```
//--9/16/200'

ln(10) =;
```

```
>>ln(10) =
LN(10)=2.302585092994
>>
```

Access Worksheet Values

You can also use the Script Window to read and write worksheet values, or to perform math operations on datasets.

1. Enter the following data into a fresh worksheet:



	A(X)	B(Y)
Long Name		
Units		
Comments		
1	1	6
2	2	7
3	3	8
4	4	9
5	5	10

2. To return the value in the first cell of the second column, type the following:

```
cell(1,2)=
```

3. Press **ENTER**. Origin returns:

```
CELL(1,2)=6
```

You can also use the column name and row number to reference cell values.

4. Type the following:

```
col(B)[1]=
```

5. Press **ENTER**. Origin returns:

```
COL(B)[1]=6
```

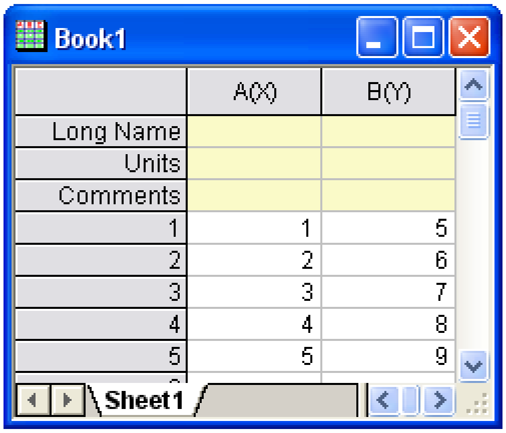
Notes: In addition to using the column name, you can also use the dataset name. In LabTalk, the syntax for naming datasets is *worksheetName_columnName*. So, For example, *Book1_A[1]*= would return the first element of column A in worksheet Data1. Also, if the worksheet that you are referencing is the active window, you can use the LabTalk string variable %H, in place of the worksheet name. For example, %H_A[1].

To subtract the value in row 1 of column A, from all the values in column B?

6. Type the following:

```
col(B)=col(B)-col(A)[1]
```

7. Press **ENTER**. Your worksheet now reads:



	A(X)	B(Y)
Long Name		
Units		
Comments		
1	1	5
2	2	6
3	3	7
4	4	8
5	5	9

Let?s use what we have learned about executing multiple lines of script in the Script window. We will multiply every value in a column of data by some constant b.

8. Type the following:

```
b=3;
```

Press **CTRL+ENTER**. Recall that this gives us a carriage return without executing the command.

9. Now type:

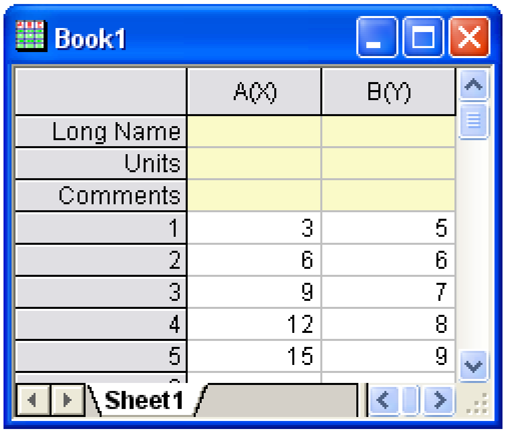
```
col(A)=col(A)*b;
```

Again, Press **CTRL+ENTER**.

10. Choose **Edit** from the Script window?s menu bar. **Script Execution** should have a check mark next to it; if not, single-click on the menu item to place a check mark there.

11. Now, select the two lines of script that you just entered into the Script window and press **ENTER**.

Your worksheet now reads:



	A(X)	B(Y)
Long Name		
Units		
Comments		
1	3	5
2	6	6
3	9	7
4	12	8
5	15	9

Notes: The following C notation is also supported:

b=3;

col(A)*=b;

You can also use linear interpolation or extrapolation on a specified *X dataset* to find the corresponding interpolated or extrapolated value in a *Y dataset*. This requires using a new notation with parentheses ( ) instead of brackets [ ].

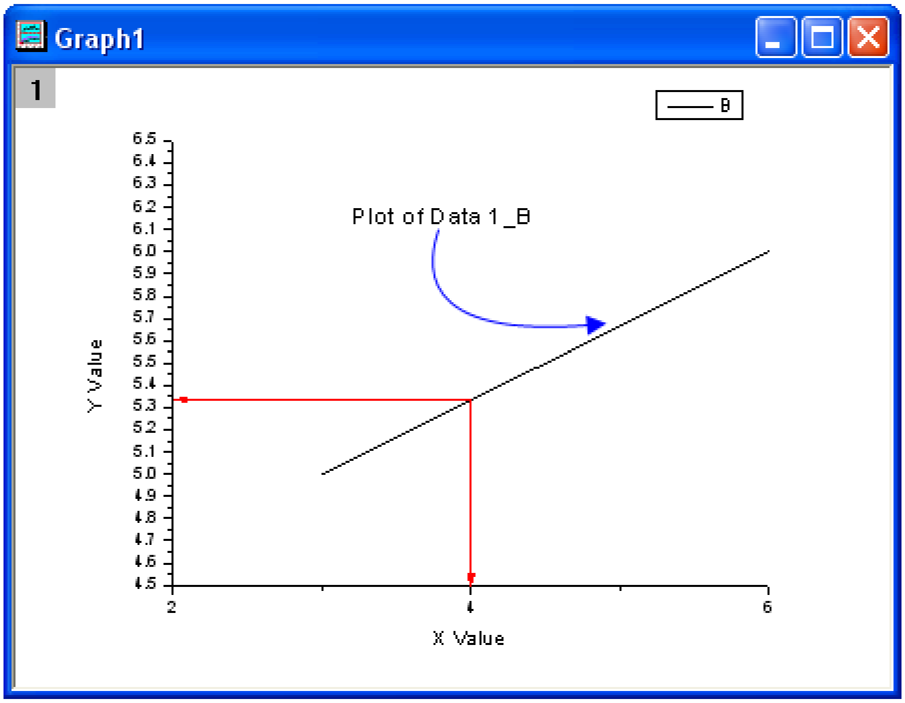
In this example, book1_b is a Y dataset and (4) is a value in an X dataset (book1_a) for which you want to find a corresponding, interpolated Y value.

12. Type the following:

```
book1_b(4) =
```

13. Press **ENTER**. Origin returns:

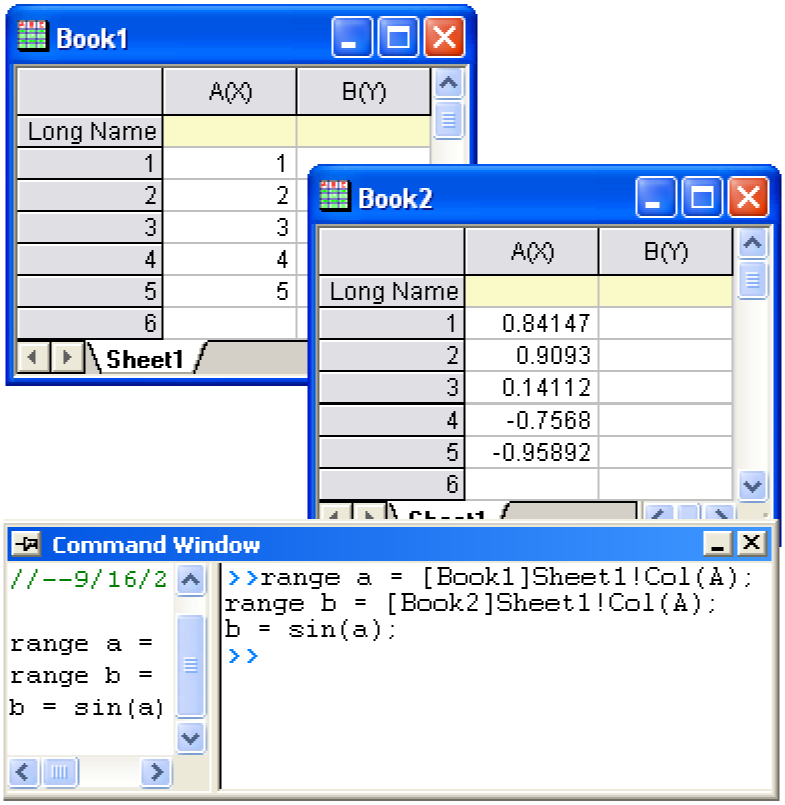
```
BOOK1_B(4)=5.333333
```



This is a line plot of our simple worksheet data. You can see that our interpolated Y value ? the one corresponding to X = 4 ? is 5.333333.

14. If the columns you work with are in different worksheet/workbook, you should use the range variables to represent the worksheet columns. For example, this script calculates the sine value on Book1 column A, and puts the result in Book2 column A (You must have Book2 before hitting Enter):

```
range a = [Book1]Sheet1!Col(A);
range b = [Book2]Sheet1!Col(A);
b = sin(a);
```



	A(X)	B(Y)
Long Name		
1	1	
2	2	
3	3	
4	4	
5	5	
6		

	A(X)	B(Y)
Long Name		
1	0.84147	
2	0.9093	
3	0.14112	
4	-0.7568	
5	-0.95892	
6		

```
//--9/16/2

range a =
range b =
b = sin(a)
```

```
>>range a = [Book1]Sheet1!Col(A);
range b = [Book2]Sheet1!Col(A);
b = sin(a);
>>
```

Access X-Functions

Origin 8 provides a large collection of X-Functions for performing a wide variety of data processing tasks. Of this collection, many of the X-Functions are accessible from LabTalk script. The functions accessible from script provide a powerful environment for users to create custom script code for their routine tasks.

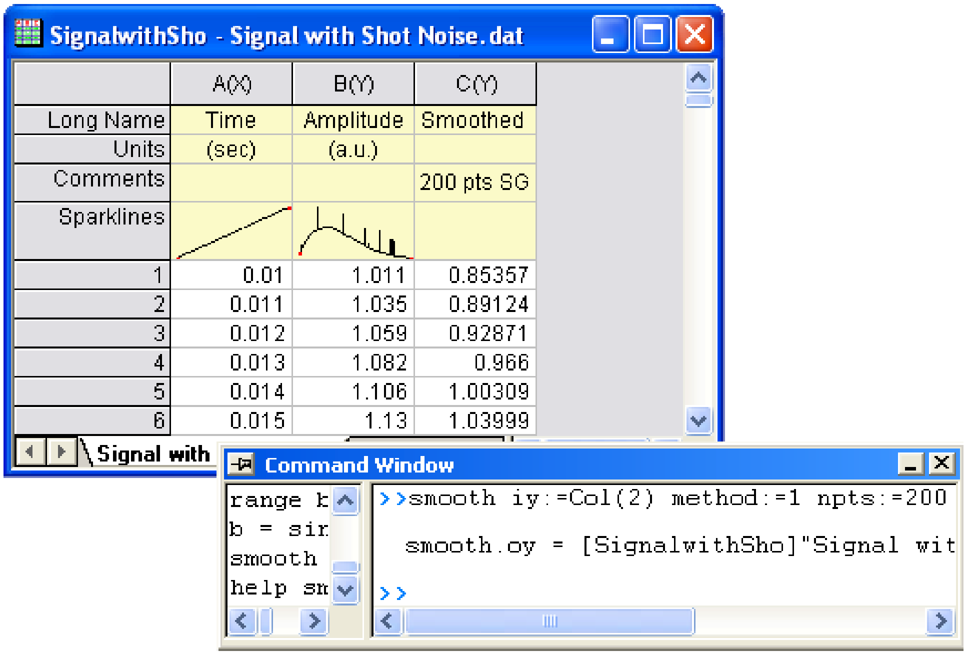
X-Functions that are accessible from script can be listed in the Command Window, and you can also obtain help on the command syntax as well as make use of auto completion of commands for such functions.

X-Functions accept data range string or range variable for specifying source and destination data for the operation. For example, the smooth X-Function under signal processing can be accessed from the Command Window as follows:

1. Import the file *Samples\Single Processing\Signal with Shot Noise.dat*.
2. In the Command Window, type the following:

```
smooth iy:=Col(2) method:=1 npts:=200
```

3. When you press **ENTER**, the result will append to the source worksheet.



SignalwithSho - Signal with Shot Noise.dat

	A(X)	B(Y)	C(Y)
Long Name	Time	Amplitude	Smoothed
Units	(sec)	(a.u.)	
Comments			200 pts SG
Sparklines			
1	0.01	1.011	0.85357
2	0.011	1.035	0.89124
3	0.012	1.059	0.92871
4	0.013	1.082	0.966
5	0.014	1.106	1.00309
6	0.015	1.13	1.03999

Command Window

```
range b
b = sir
smooth
help sm
```

```
>>smooth iy:=Col(2) method:=1 npts:=200

  smooth.oy = [SignalwithSho]"Signal wit

>>
```

4. For help in using this smooth X-Function, you can type

```
help smooth
```

5. to open the corresponding Help.

11.2 Introduction to Origin C and Code Builder

11.2.1 Summary

Origin C supports a nearly-complete ANSI C language syntax as well as a subset of C++ features including internal and DLL-extended classes. In addition, Origin C is ?Origin aware?. This means that Origin objects such as worksheets and graphs are mapped to classes in Origin C, allowing direct manipulation of these objects and their properties from Origin C.

Origin C?s integrated development environment (IDE) is called Code Builder. Code Builder provides standard tools for writing, compiling, and debugging your Origin C programs. Once an Origin C function is compiled, the function can be called in various ways from the Origin or Code Builder workspaces.

Minimum Origin Version Required: Origin 8.0 SR0

11.2.2 What you will learn

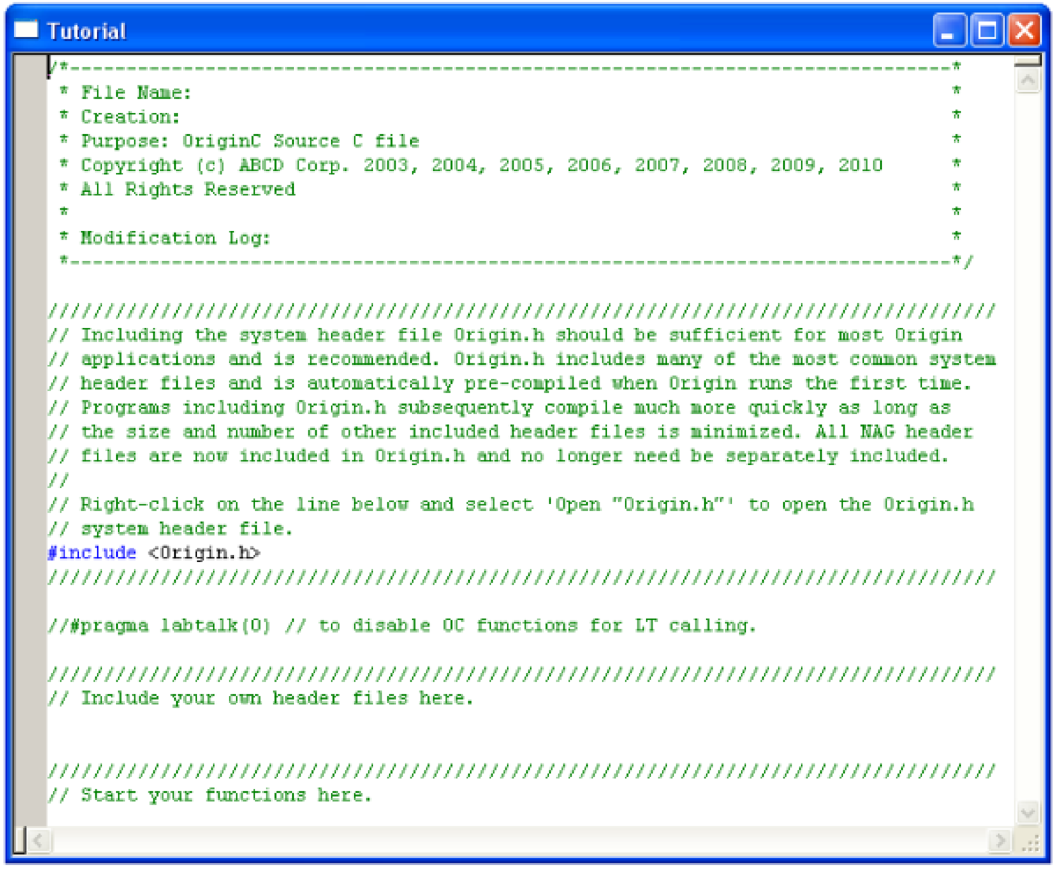
This tutorial introduces you to Origin C and Code Builder by showing you how to write, compile and call a function that types the message ?Hello World!!!?.

11.2.3 Steps

1. On the Origin Standard toolbar, click the **Code Builder** button

2. On the Code Builder toolbar, click the **New** button

   In the **New File** dialog, select **C File**.
   In the **File Name** text box, type **Tutorial**.
   In the **Location** text box, select the \Origin C subfolder (Note: The Browse button looks like this:...).
   Click **OK**. A file named Tutorial.c opens in the Code Builder workspace.



```
/*------------------------------------------------------------------------------*
 * File Name:                                                                   *
 * Creation:                                                                    *
 * Purpose: OriginC Source C file                                               *
 * Copyright (c) ABCD Corp. 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010      *
 * All Rights Reserved                                                          *
 *                                                                              *
 * Modification Log:                                                            *
 *------------------------------------------------------------------------------*/

////////////////////////////////////////////////////////////////////////////////////
// Including the system header file Origin.h should be sufficient for most Origin
// applications and is recommended. Origin.h includes many of the most common system
// header files and is automatically pre-compiled when Origin runs the first time.
// Programs including Origin.h subsequently compile much more quickly as long as
// the size and number of other included header files is minimized. All NAG header
// files are now included in Origin.h and no longer need be separately included.
//
// Right-click on the line below and select 'Open "Origin.h"' to open the Origin.h
// system header file.
#include <Origin.h>
////////////////////////////////////////////////////////////////////////////////////

//#pragma labtalk(0) // to disable OC functions for LT calling.

////////////////////////////////////////////////////////////////////////////////////
// Include your own header files here.


////////////////////////////////////////////////////////////////////////////////////
// Start your functions here.
```

3. Type the following beneath the line that reads // start your functions here:

```
void test ()
{
   printf("Hello World!!!\n");
}
```

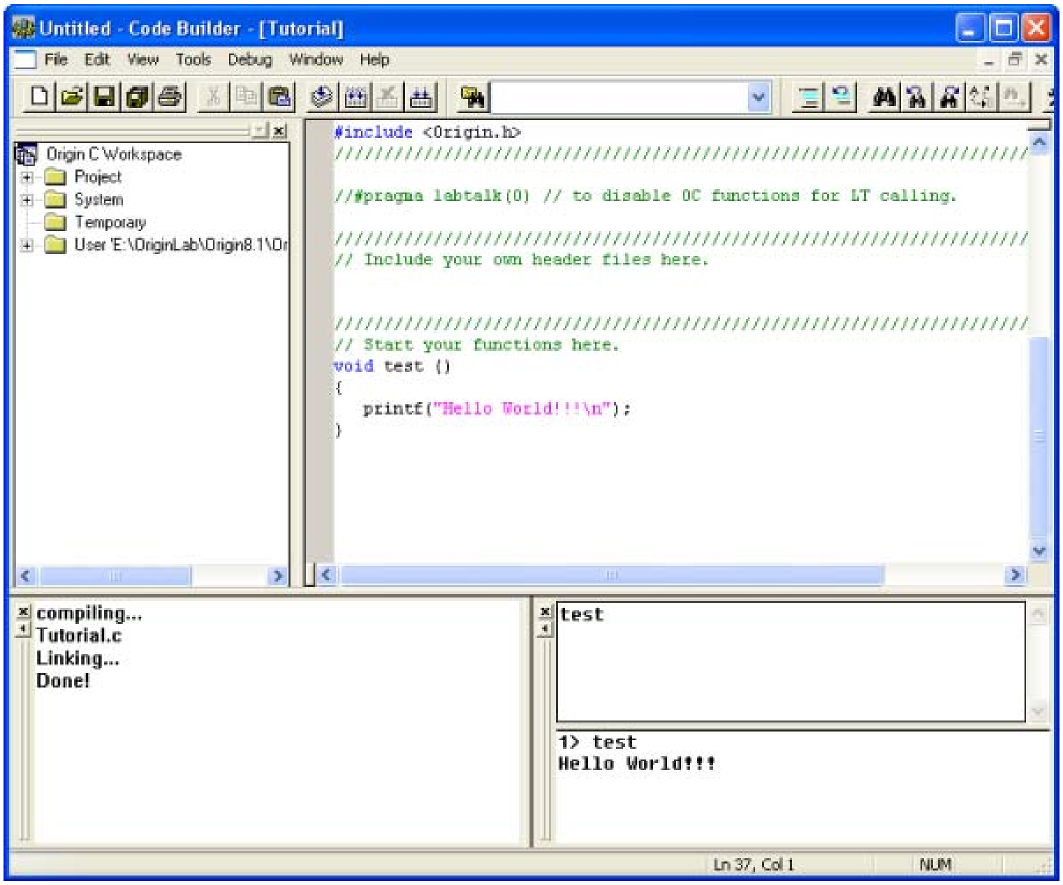
4. On the Code Builder workspace Standard toolbar, click the **Build** button. This compiles the test function.

5. To call this function, click in the upper pane of the **LabTalk Console**. This is located in the lower right corner of the **Code Builder** workspace (This is the default location. If the LabTalk console isn?t visible, select **View: LabTalk Console** from the Code Builder menu and make sure that the menu item is checked).

6. Type the following LabTalk function call in the **LabTalk Console**:

```
test
```

7. Press **ENTER**.



Untitled - Code Builder - [Tutorial]
File Edit View Tools Debug Window Help
Origin C Workspace
Project
System
Temporary
User 'E:\OriginLab\Origin8.1\Or
Ln 37, Col 1
NUM

```
#include <Origin.h>
////////////////////////////////////////////////////////////////////////////

//#pragma labtalk(0) // to disable OC functions for LT calling.

////////////////////////////////////////////////////////////////////////////
// Include your own header files here.


////////////////////////////////////////////////////////////////////////////
// Start your functions here.
void test ()
{
   printf("Hello World!!!\n");
}
```

```
compiling...
Tutorial.c
Linking...
Done!
```

```
test
```

```
1> test
Hello World!!!
```

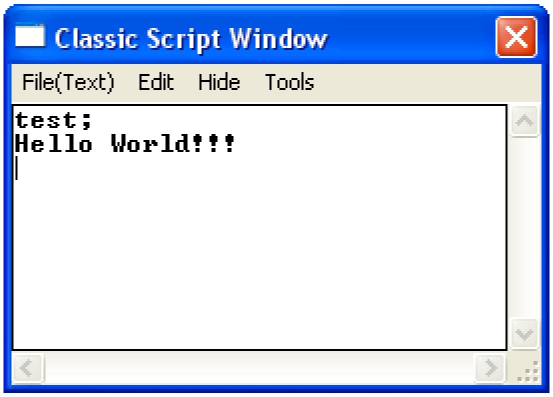
To test this function in the Origin Script Window:

8. Return to the Origin workspace, and select **Window: Script Window**.

9. In the Script window, type the following:

```
test
```

10. Press **ENTER**.



```
test;
Hello World!!!
```

"**Hello World!!!**" displays in the Script Window.

This concludes the **Introduction to Origin C and Code Builder** tutorial.

11.3 Adding New Origin C Functions to Origin

11.3.1 Summary

Functions written in Origin C are accessible from various locations within the Origin interface, such as the Script Window, provided they meet the following criteria: the function should return either void (as in the previous tutorial), double, string, or vectors of type double or string. Variables passed to the function from Origin should be of type double or string, or vectors of these types. Functions that do not meet these criteria are not callable from the Origin interface, but can be called within other Origin C functions. Note that although an Origin C function that accepts and returns type int can be called from the Origin interface, the data may be truncated since the interface only supports type double.

In this tutorial, you will be introduced to writing a math function that returns computed values. We will first create a function that returns type double to Origin, and then we will create a function that returns vectors of type double.

Minimum Origin Version Required: Origin 8.0 SR0

11.3.2 What you will learn

How to add a new function and how to run this function in the Script Window.

11.3.3 Steps

1. Start a new Origin C file in **Code Builder**.
2. Enter the following code:

```
double myfunc1(double x, double a)
{
         return sin( a * x );
}
```

3. Click the **Build** button to compile the function.

This function can now be called from the Origin interface, in places such as the Script Window.

4. Go to the Script Window, and type in the following lines, pressing **ENTER** at the end of each line:

```
y = myfunc1(2, 3)
y =
```

You can also use worksheet cells instead of absolute numbers:

5. Make a worksheet active, enter a number in the first row of column A. Then type the following into the Script Window and press **ENTER**:

```
col(B)[1] = myfunc1(col(A)[1], 3)
```

Note that a function such as myfunc1, that accepts and returns type double, can also be used to perform vector operations.

6. Fill rows 1 through 10 of Column A with numbers, and type the following into the Script Window:

```
col(B) = myfunc1(col(A), 3)
```

In the above example, Origin calls the myfunc1 function for each row of column A. For performing vector operations as above, it is more efficient to write functions that accept and return vectors.

7. Go back to Code Builder and add the following function to the same file, and compile the file by clicking the **Build** button

```
vector<double> myfunc2(vector<double>& vecIn, double a)
{
       vector<double> vecOut;
       vecOut = sin( a * vecIn );
       return vecOut;
}
```

8. Go back to the Origin interface, fill Column A with some new numbers, and type the following into the Script Window:

```
col(B) = myfunc2(col(A), 3)
```

The function myfunc2 is called only once for computing the entire column.

Note that you can use such functions in other places such as the ?Set Column Values? dialog. The Auto Update feature of ?Set Column Values? can be enabled by checking the appropriate check box in this dialog. As long as the Origin C function is compiled and ready to be called from Origin, any changes to the source column will result in an update of the destination column.

11.4 The Code Builder Workspace

11.4.1 Summary

In this exercise, we will create a workspace, add a source file with a new function, then build, test and save the workspace file.

Minimum Origin Version Required: Origin 8.0 SR0

11.4.2 What you will learn

This tutorial will show you how to:

- Build a Workspace File

- Build Workspace Folders
- Build on Startup

11.4.3 The Workspace File

A workspace is a collection of files that can be opened by a single menu option (**File: Open Workspace?**) in Origin?s Code Builder. Any text file can be a part of the collection. They do not necessarily have to be source code files; they could be notes, for example.

All files opened in the Multiple Document Window by a workspace can be edited and saved individually. In addition to files being opened in the Multiple Document Window, source code files can be added to the Workspace Window with the **File: Add to Workspace** menu option.

By including source code files in the Workspace Window, you can build individual or multiple files with the appropriate menu option or toolbar button. Header files can be referenced within source files and do not need to be loaded in the Workspace Window or even open in the Multiple Document Window.

Since you can save a workspace with a new name, you can have multiple workspace files. However, only one workspace file can be open at a time.

To create a workspace:

1. On the **Standard** toolbar, click the **Code Builder** button .
2. From the Code Builder menu, select **File: New Workspace**. This creates a new workspace with the default name of ?Untitled.ocw?.
3. From the Code Builder menu, select **File: New**. This opens the **New** File dialog.
4. Choose **C File**, and type **foo** in the **File Name** text box. The **Add to Workspace** and **Fill with Default Contents** check boxes should be selected. You may accept the default Location. Click **OK**.
5. In FOO.C, starting below the line that says ?\\start your functions here?, type the following:

```
void bar()
{
        printf("Hello World!\n");
}
```

6. Click the **Build** button . Origin automatically saves the source file and compiles and links the function.

7. From the Origin menu choose **Window: Script Window**.

8. To test our new function, type:

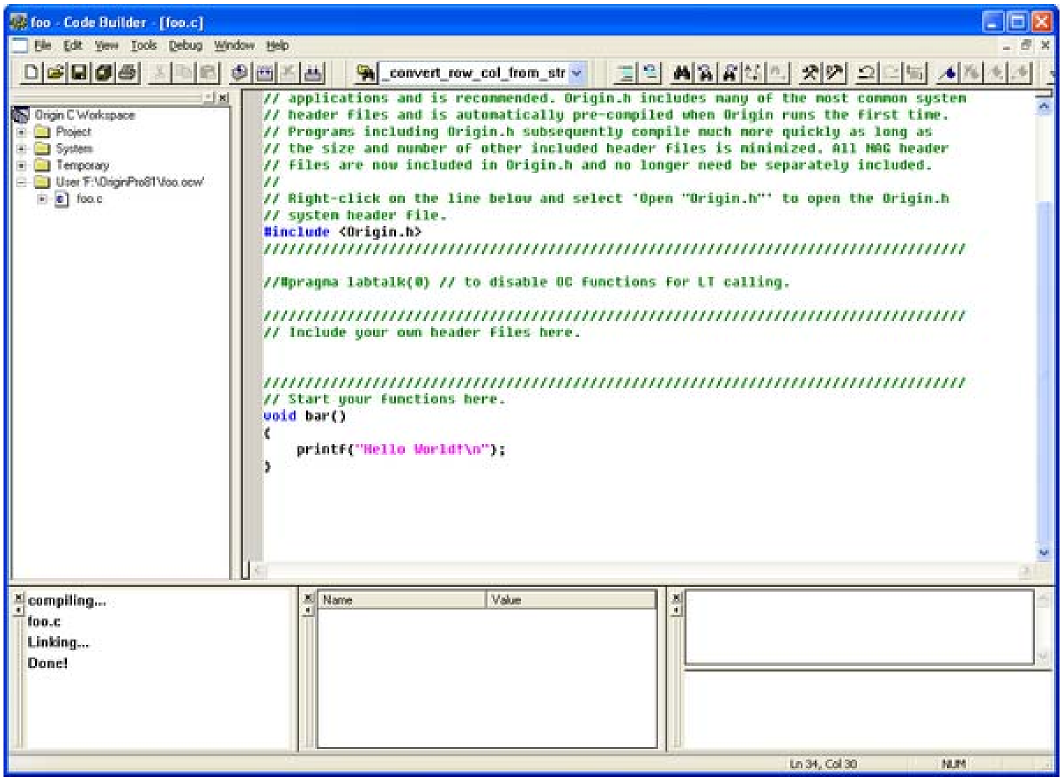
```
bar
```

9. Press **ENTER**

Origin responds by typing **Hello World!!!**

10. From the Code Builder menu, choose **File: Save Workspace As?.**

The figure shows the foo.ocw workspace file containing a single source file, foo.c, in the Multiple Document Window. The file has been added to the Workspace Window. The Output Window shows that the file has been compiled. The source file contains a single function ? bar( ) ? which is listed in the tree structure of the workspace.


foo - Code Builder - [foo.c]
File Edit View Tools Debug Window Help
_convert_row_col_from_str
Origin C Workspace
Project
System
Temporary
User F:\OriginPro81\foo.ocw
foo.c
Name
Value
Ln 34, Col 30
NUM

```
// applications and is recommended. Origin.h includes many of the most common system
// header files and is automatically pre-compiled when Origin runs the first time.
// Programs including Origin.h subsequently compile much more quickly as long as
// the size and number of other included header files is minimized. All NAG header
// files are now included in Origin.h and no longer need be separately included.
//
// Right-click on the line below and select 'Open "Origin.h"' to open the Origin.h
// system header file.
#include <Origin.h>
////////////////////////////////////////////////////////////////////////////////////

//#pragma labtalk(0) // to disable OC functions for LT calling.

////////////////////////////////////////////////////////////////////////////////////
// Include your own header files here.


////////////////////////////////////////////////////////////////////////////////////
// Start your functions here.
void bar()
{
    printf("Hello World!\n");
}
```

```
compiling...
foo.c
Linking...
Done!
```

11.4.4 Workspace Folders

The Code Builder workspace has four subfolders named Project, System, Temporary, and User. Files added by user, such as foo.c that you just added, are placed in the User subfolder. Origin, itself, uses Origin C for many analysis routines. When these routines are accessed, the Origin C source files are loaded into the Workspace into either the System subfolder or the Temporary subfolder. The Project subfolder is reserved for files that are saved and loaded with the Origin project. This aspect of attaching a file to the project is described in a separate tutorial.

11.4.5 Build on Startup

If you right click on "Origin C Workspace", the shortcut menu has a **Build on Startup** option. When this option is checked, the last workspace you saved will be loaded when you restart Origin. All source files in the Workspace will be built and all functions in the source files will be available for immediate use.

For information on building individual source files on startup by including information in the ORIGIN.INI file, please view Build on Startup

This concludes the tutorial on the **Code Builder Workspace**.

11.5 Organizing and Accessing Origin C Functions

11.5.1 Summary

Techniques for using your Origin C functions.

Minimum Origin Version Required: Origin 8.0 SR0

11.5.2 What you will learn

This tutorial will show you how to:

- Save your Origin C Function with your Project
- Associate your Programs with Visual Objects
- Load and Compile your Origin C Function from script

11.5.3 Saving your Origin C Function with your Project

One way to load and compile your Origin C function is to save the Origin C file as an attachment to your Origin Project (*.OPJ) file. When a project file is opened, all files attached to it are separated out and stored in a temporary folder. In addition, any Origin C file that was attached is also automatically loaded into the Code Builder workspace, and compiled. The function is then ready to be called upon opening the Origin Project file.

1. Start a new Origin project file by clicking on the **New** Project button on the **Standard** Toolbar.
2. On the **Standard** Toolbar, click the **Code Builder** button .
3. From the Code Builder menu, select **File: New**. This opens the **New File** dialog box.
4. In the top list-box, select **C File**.
5. In the **File Name** text box, type: **Test**. Keep the **Add to Workspace** check box selected. Click **OK**. The file Test.c is added to the workspace.
6. Select and copy the following function, and paste it into the Test.c file. Be sure to paste the text below the line that reads ?//start your functions here.?

```
void Plot_Data(string strTemplate, string strData)
{
        // Create a graph window from a Template
        GraphPage gp;
        BOOL bOK = gp.Create(strTemplate, CREATE_VISIBLE);
        if( !bOK )
                return;

        // Attach the first layer (0) to a GraphLayer object
        GraphLayer gl = gp.Layers(0);

        //Attach a dataset to a Curve object
        Curve crv(strData);

        // Add the Curve to the graph layer
        int nPlot = gl.AddPlot(crv);
        if(nPlot >= 0)
        {
                // Set plot color to Green(2)
                gl.DataPlots(nPlot).SetColor(2, TRUE);

                // Rescale this graph layer
                gl.Rescale();
        }
}
```

The Plot_Data function takes two arguments: (1) the template name and (2) the name of a Y dataset to include (plot) in the layer.

7. Click the **Build** button to compile and link the file.

8. Drag-and-drop the file Test.c from the User subfolder branch of the Workspace tree, to the Project subfolder. (**Hint:** You may need to first expand the User subfolder branch to display the Test.c entry prior to dragging the file).

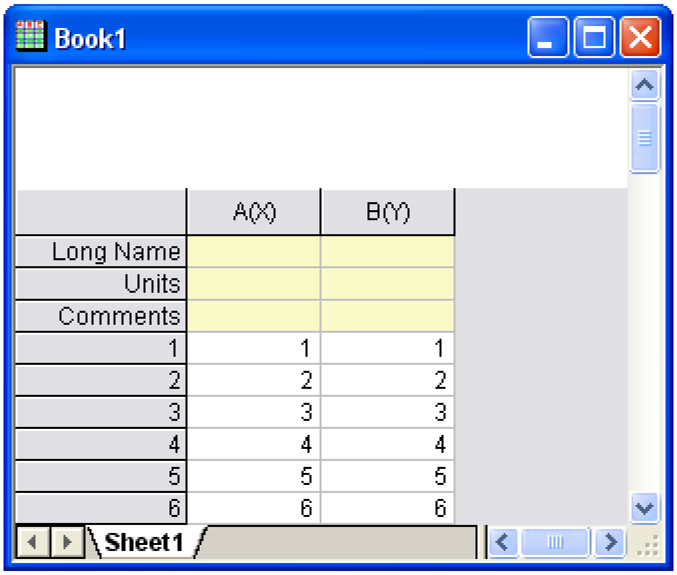
9. Go back to the Origin interface and save the project by clicking the **Save** button on the Standard Toolbar. Give the project the name Test.OPJ, and save it in a location of your choosing.

10. The Origin C file, Test.c, is now saved with the Project. To verify this, close the project, and go back to Code Builder. You will see that there are no entries under the Project subfolder of the Workspace tree. Now go back to Origin interface and reopen the project. Go to Code builder and verify that Test.c is now listed under the Project subfolder (**Hint:** you may need to expand the Project subfolder branch to see the Test.c entry).

11.5.4 Associating your Programs with Visual Objects

You will now learn how to create a button on a worksheet and program the button to call the Origin C function in the Test.c file that you saved with the project.

1. Open the Test.OPJ project that you saved under step 9 (previous section).
2. Highlight the A(X) and B(Y) columns, right-click and select **Fill Columns With: Row Numbers.**
3. From the menu, select **Format: Worksheet** to open the **Worksheet Properties** dialog box.
4. In the **Size** tab, **Worksheet Measurement** branch, set the **Gap from Top** to **40** and click **OK** to close the dialog.



	A(X)	B(Y)
Long Name		
Units		
Comments		
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6

The worksheet now has sufficient space above the column headings to add a text label.

Right-click in the area directly above the two columns and choose **Add Text**.

5. At the cursor, type: **Plot Data**.

6. Click once outside the text label to deselect it.

7. Right-click on the text label and choose **Programming Control** to open the **Programming Control** dialog box. (**Hint:** Please choose **Label Control** in Origin 7.5)

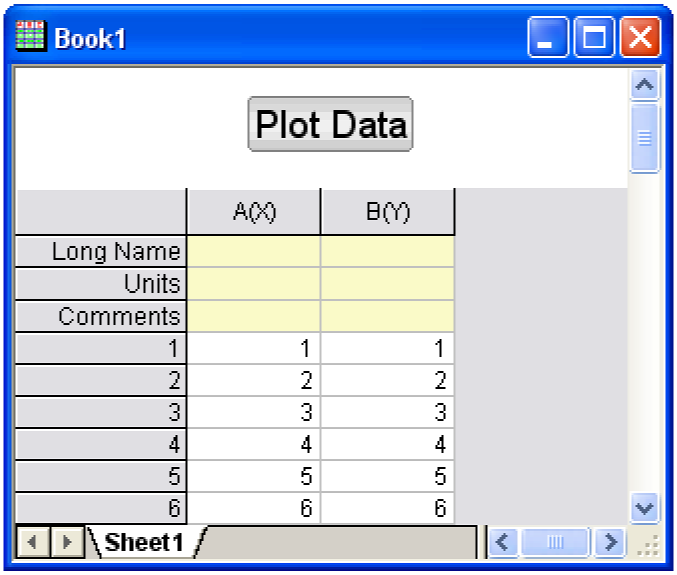
8. From the **Script, Run After** drop-down list, choose **Button Up**.

9. Type the following script in the text box at the bottom of the dialog box:

```
Plot_Data("scatter","book1_b");
```

10. Click **OK**.

Your text label will now look like a button.



	A(X)	B(Y)
Long Name		
Units		
Comments		
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6

11. Click the **Plot Data** button on your worksheet.

12. The Plot_Data function in your Test.c file is called, and a scatter plot is created.

Notes: The script behind the button assumes that you have data in column B(Y) of the Data1 worksheet and that there is an associated X data set.

11.5.5 Loading and Compiling your Origin C Function from Script

In this tutorial we learned how to save an Origin C function along with the project file and then access the function from the Origin interface. Saving an Origin C file with a project limits access to functions within that file to only that project. When a new project is opened, the functions are not available any more.

To access functions in an Origin C file that is saved on disk, the file can be programmatically loaded and compiled using LabTalk script. The script command for performing the programmatic load and compile is run.LoadOC. Refer to the **LabTalk** Help files (**Help: Programming: Labtalk**) for more information on using this command.

This concludes the **Origin C Functions** tutorial.

11.6 Accessing Internal Origin Objects by Origin C

11.6.1 Summary

Internal Origin objects (such as Project Explorer folders, Origin windows (pages), layers, plots, graphic objects, data sets, etc.) are accessed using Origin C classes. To access or programmatically control an internal Origin object, you must attach it to an Origin C object.

To attach something to an internal Origin object you must first ?find? it using the properties, methods, and collections of a container class. Common container classes include the Project, Folder, Page, GraphPage, Layer, GraphLayer, Worksheet, MatrixLayer, and Collection classes. Once found, an internal Origin object can easily be attached to an Origin C object of the appropriate type.

The internal Origin object is then programmatically controlled by manipulating the class methods and properties of the attached Origin C object. The objective of this tutorial is to demonstrate how to find particular internal Origin objects, attach things to those objects, and access the objects by manipulating the methods and properties of the attached Origin C objects.

Minimum Origin Version Required: Origin 8.1 SR1

11.6.2 What you will learn

This tutorial will show you how to:

- Access Worksheet Related Objects
- Access Graph Related Objects

11.6.3 Accessing Worksheet Related Objects

Familiarity with the Origin C Project class (Project.h), the Collection class (Collection.h), and the Folder class (Folder.h), is valuable when attempting to understand how to find particular internal Origin Objects. Users may find it helpful to preview these classes in the **Origin C Reference: Classes** book of **Origin C Help** or in the above header files located in the ..\Origin\OriginC\system subfolder. If you are not familiar with debugging Origin C files in Code Builder you may also find it helpful to review the Debug Tutorial before proceeding.

To begin this tutorial:

1. On the **Standard** toolbar, click the **New Project** button .
2. On the **Standard** toolbar, click the **Code Builder** button .
3. On the **Code Builder** menu, select **File: New Workspace**.
4. On the **Code Builder** menu, click the **Open** button .
5. Browse to the \Samples\Origin C Examples\Programming Guide\Introduction to Accessing Origin Objects folder in the Origin software directory, select AccessWorksheetObjectsTutorial.c, check the **Add to Workspace** check box and click **Open**.
6. On the **Code Builder** toolbar, click the **Rebuild All** button . This compiles and links the file.
7. On the Code Builder **View** menu, verify that the **LabTalk Console** (Command & Results) and the **Local Variables** windows are visible (the corresponding menu items should be checked).
8. From the **Code Builder** menu, select **Tools: Customize**. Select the Toolbars tab and make sure that the **Debug** toolbar check box is selected.
9. In Code Builder, activate AccessWorksheetObjectsTutorial.c.
10. Near the top of the file, locate and click on the line:

```
PageBase pb;
```

You can position the cursor anywhere on the line.

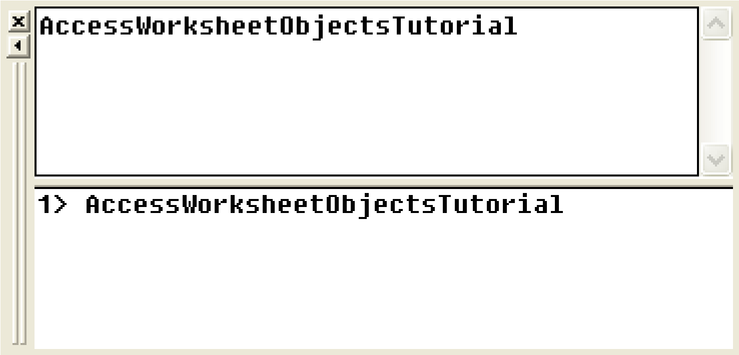
11. From the Code Builder menu, select **Debug: Toggle Breakpoints**. Alternately, press **F9** or click the **Toggle Breakpoint** button on the **Debug** toolbar.

A brown circle is displayed in the gray margin to the left of the above line indicating that a Debug breakpoint has been set for that line.

12. In the Code Builder workspace, activate the LabTalk Console (Command & Results window) and type in the following:

```
AccessWorksheetObjectsTutorial
```

13. Press **ENTER** to execute the function.

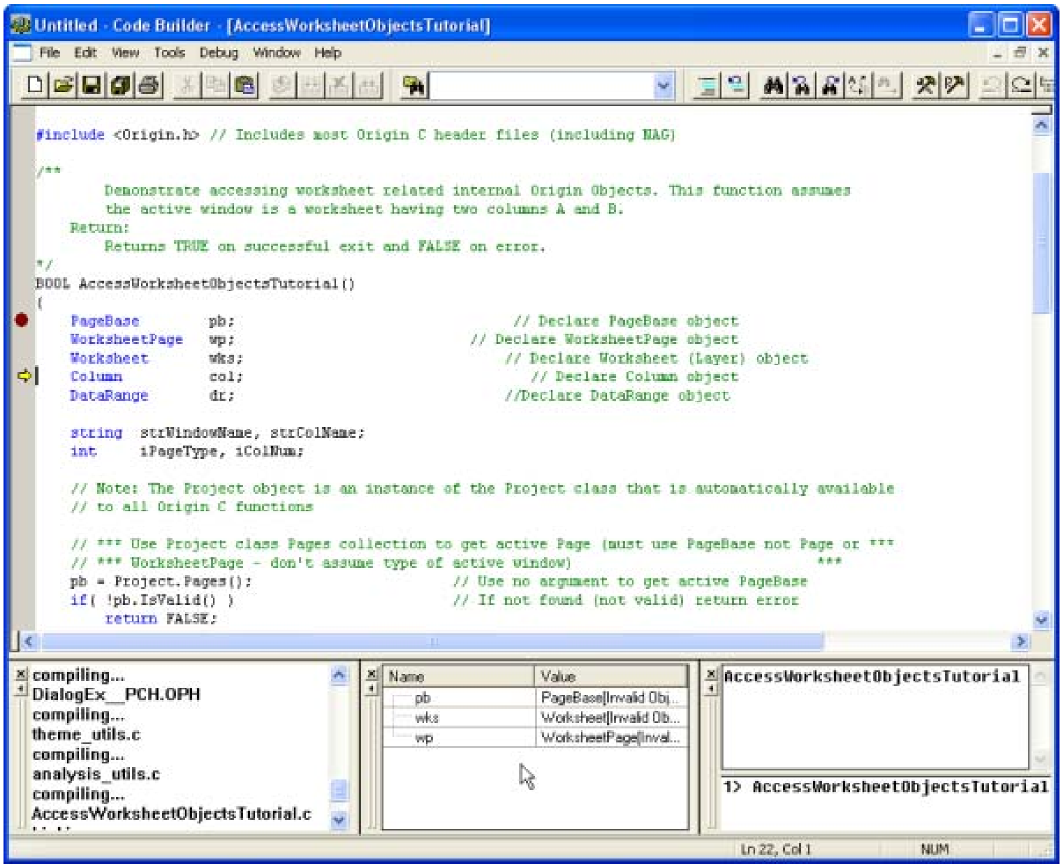


```
AccessWorksheetObjectsTutorial
```

```
1> AccessWorksheetObjectsTutorial
```

14. On the **Debug** toolbar, press the **Step Into** button

15. Press the **Step Into** button repeatedly, stopping to read the comments for each statement. Periodically stop and resize and/or reposition the Local Variables window to view the current run-time value of each variable.


Untitled - Code Builder - [AccessWorksheetObjectsTutorial]
File Edit View Tools Debug Window Help

```
#include <Origin.h> // Includes most Origin C header files (including NAG)

/**
        Demonstrate accessing worksheet related internal Origin Objects. This function assumes
        the active window is a worksheet having two columns A and B.
    Return:
        Returns TRUE on successful exit and FALSE on error.
*/
BOOL AccessWorksheetObjectsTutorial()
{
    PageBase        pb;                             // Declare PageBase object
    WorksheetPage   wp;                        // Declare WorksheetPage object
    Worksheet       wks;                           // Declare Worksheet (Layer) object
    Column          col;                              // Declare Column object
    DataRange       dr;                            //Declare DataRange object

    string  strWindowName, strColName;
    int     iPageType, iColNum;

    // Note: The Project object is an instance of the Project class that is automatically available
    // to all Origin C functions

    // *** Use Project class Pages collection to get active Page (must use PageBase not Page or ***
    // *** WorksheetPage - don't assume type of active window)                                 ***
    pb = Project.Pages();                       // Use no argument to get active PageBase
    if( !pb.IsValid() )                         // If not found (not valid) return error
        return FALSE;
```

```
compiling...
DialogEx__PCH.OPH
compiling...
theme_utils.c
compiling...
analysis_utils.c
compiling...
AccessWorksheetObjectsTutorial.c
```

Name	Value
pb	PageBase[Invalid Obj...
wks	Worksheet[Invalid Ob...
wp	WorksheetPage[Inval...

```
AccessWorksheetObjectsTutorial
```

```
1> AccessWorksheetObjectsTutorial
```

11.6.4 Accessing Graph Related Objects

1. Return to the *Origin* workspace and, on the **Standard** toolbar, click the **Open** button .
2. Browse to the \Samples\Origin C Examples\Programming Guide\Introduction to Accessing Origin Objects subfolder, select AccessGraphObjectsTutorial.OPJ, and click **Open**. You may be prompted to save changes to an untitled project. Click **No** and a worksheet and graph should open.
3. From the Code Builder menu, select **File: New Workspace**. Click **No** when prompted to save workspace changes.
4. In Code Builder, click the **Open** button .
5. Browse to the \Samples\Origin C Examples\Programming Guide\Introduction to Accessing Origin Objects subfolder, select AccessGraphObjectsTutorial.c, select the **Add to Workspace** check box, and click **Open**.
6. Click the **Rebuild All** button to compile and link the file.
7. On the Code Builder **View** menu, verify that the **LabTalk Console** (Command & Results) and the **Local Variables** windows are visible (the corresponding menu items should be checked).
8. From the Code Builder menu, select **Tools: Customize**. Select the Toolbars tab and make sure that the **Debug** toolbar check box is selected.
9. In the Code Builder workspace, activate AccessGraphObjectsTutorial.c.
10. Near the top of the file locate and click on the line:

```
GraphPage gp;
```

You can position the cursor anywhere on the line.

11. From the Code Builder menu, select **Debug: Toggle Breakpoints**. Alternately, press **F9** or click the **Toggle Breakpoint** button on the **Debug** toolbar.

A brown circle is displayed in the gray margin to the left of the above line indicating that a Debug breakpoint has been set for that line.

12. Activate the LabTalk Console (Command & Results window) in Code Builder and type the following:

```
AccessGraphObjectsTutorial
```

13. Press **ENTER** to execute the function.

14. On the **Debug** toolbar, press the **Step Into** button

15. Press the **Step Into** button repeatedly, stopping to read the comments for each statement. Periodically stop and resize and/or reposition the Local Variables window to view the current run-time value of each variable.

This concludes the **Internal Origin Objects** tutorial.

11.7 Advanced Scripting using LabTalk

11.7.1 Summary

This tutorial demonstrates how to use some advanced LabTalk scripting commands and methods to organize your script files. To learn more about all the commands and methods supported in LabTalk, please refer to **Help: Programming: LabTalk**.

Minimum Origin Version Required: Origin 8.1 SR1

11.7.2 What you will learn

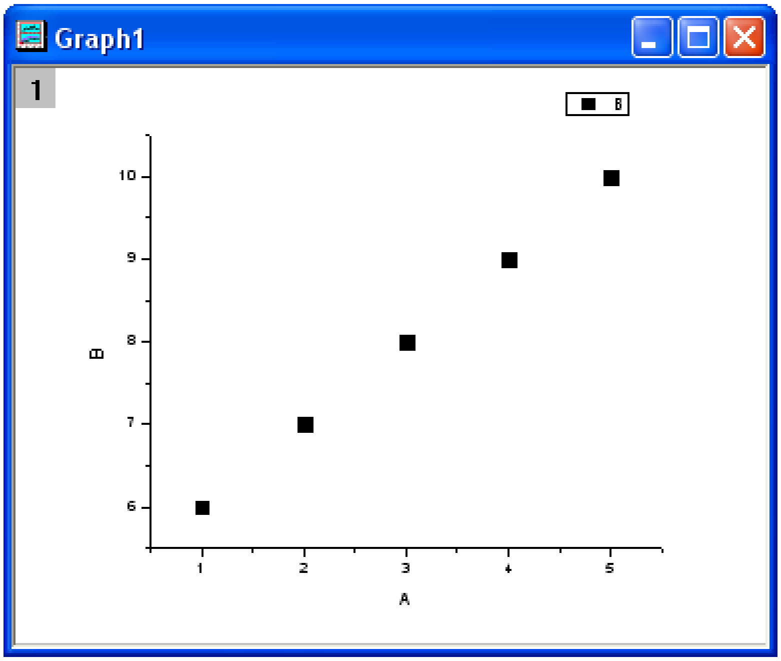
This tutorial will show you how to:

- Modify Plot Attributes via Script
- Define a LabTalk Script Macro Command
- Define a Macro
- See Origin?s Predefined System Macros
- Load and Compile your Origin C Function using LabTalk script
- Use .OGS Files to Store Script.

11.7.3 Modifying Plot Attributes via Script

This section demonstrates how to use script commands to change the attributes of a data plot.

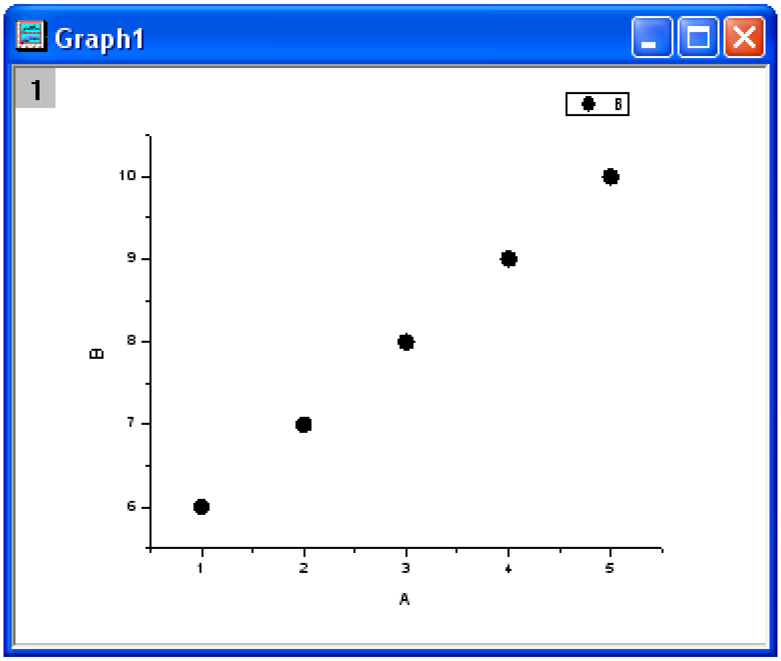
1. Start a new project, enter the numbers 1 through 5 in column A of the Data1 worksheet and numbers 6 through 10 in column B. If you have not already done so, open the Script Window by selecting **Window: Script Window** from the Origin program menu.
2. Using the worksheet data from the previous exercise, create a scatter plot. Note that the scatter plot symbol is a black filled square (symbol size is increased for clarity).



3. To change the symbol shape, type the following:

```
set %C -k 2
```

4. Press **ENTER**.

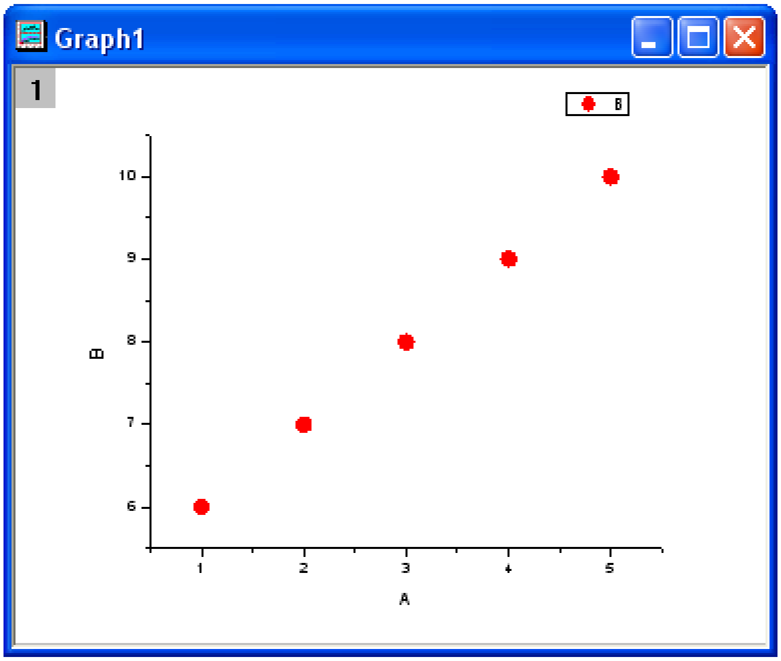


The data plot symbol changes from a filled square to a filled circle (the numbers correspond to Origin's symbol list; 1 = square, 2 = circle, etc.).

5. To change the symbol color, type the following:

```
set %C -c 2
```

6. Press **ENTER**.



The data plot symbol color changes from black to red.

To modify the axis scale values

7. Type the following:

```
X1=0;X2=20;Y1=0;Y2=10
```

8. Press **ENTER**.

Your X-axis scale now reads from 0-20, and the Y-axis reads from 0-10.

Notes: As this example illustrates, you can type multiple lines of script in a single line by separating commands with a semi-colon.

You can also use the set command to specify the data display range.

9. Type the following:

```
Set %C -b 2
```

10. Press **ENTER**.

The graph?s display range now begins with the second data point in the data set.

11. Type the following:

```
Set %C -e 4
```

12. Press **ENTER**.

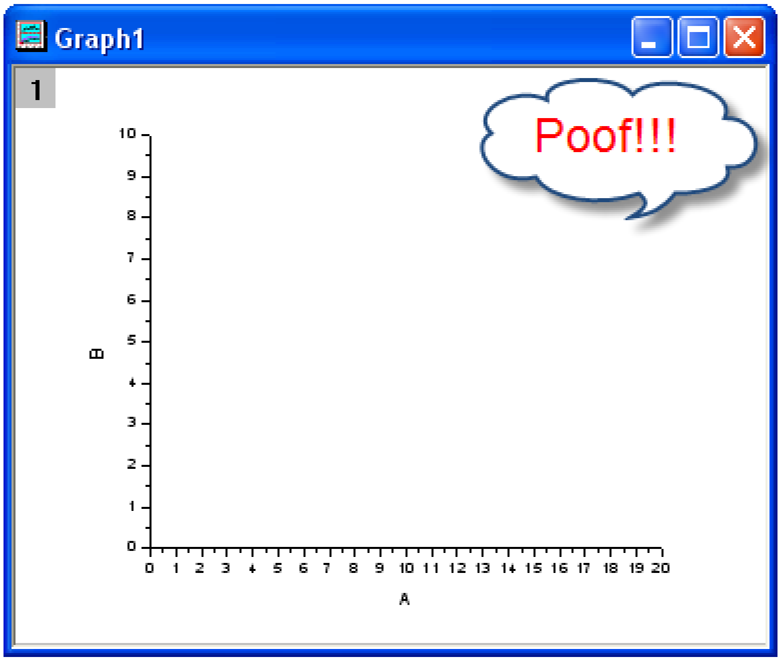
The graph?s display range now ends with the fourth data point in the data set.

You can also hide or show a data plot using the set command?s -s switch.

13. Type the following:

```
Set %C -s 0
```

14. Press **ENTER**.



The active data set is now hidden.

15. To show the hidden data set, type:

```
Set %C -s 1
```

16. Press **ENTER**.

11.7.4 Defining a LabTalk Script Macro Command

A macro is a convenient method of aliasing a LabTalk script. When you define a macro you are associating an entire script with a specific name. This name becomes a command that invokes the associated script.

When developing scripts, macros can provide several advantages.

- Modular code can streamline a script by replacing repetitive or similar blocks of code with multiple calls of the same macro.
- Modifications to your code become easier to implement because you only have to redefine your macro as opposed to modifying repeated blocks of code that are scattered throughout your application.
- There is a limit to the number of tokens that can be included between a set of curly braces that enclose script. Macros provide a means to shorten the code between braces by calling on a pre-defined macro.
- You can modify the behavior of a LabTalk command by creating a macro of the same name. The functionality of the LabTalk command is restored when the macro is deleted.

11.7.5 Defining a Macro

A macro is defined using the define command. The general syntax is:

```
define macroName {
        script
}
```

where *macroName* and *script* are the name of the macro and the body of the macro, respectively.

To define a macro using LabTalk?s define command:

1. From the Origin menu, select **Window: Script Window**.
2. Type the following:

```
def hello {
        type -b "Hello World!!!";
}
```

This script defines a macro named hello that will type "Hello World!!!".

Notes: The define command can be abbreviated as def

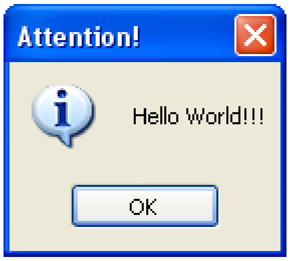
We will now use the Script Window to call our hello macro.

3. Type the following into the Script window:

```
hello
```

4. Highlight all codes in the Script Window and press **ENTER**.

An attention dialog opens to say "Hello World!!!".



Hello World!!!

11.7.6 Origin?s Predefined System Macros

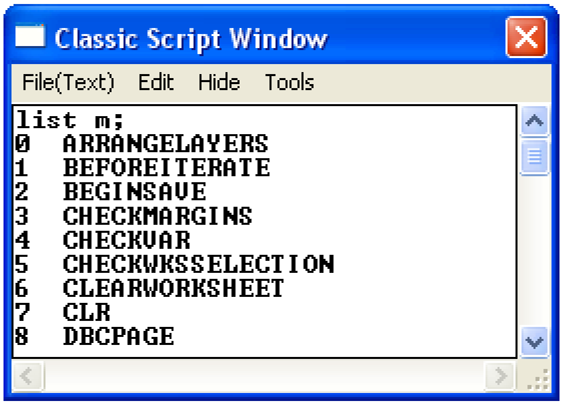
Let's look at Origin's predefined system macros, some of which take in arguments. Macros can take up to five arguments. Use the %1, %2,? %5 notation within the script definition to indicate that the macro expects one or more arguments (%1= 1st argument, %2= 2nd argument,?%5 = 5th argument).

1. In the Script Window, type the following:

```
list m
```

2. Press **ENTER**.

Origin responds by typing the names of predefined macros into the Script Window.



```
list m;
0   ARRANGELAYERS
1   BEFOREITERATE
2   BEGINSAVE
3   CHECKMARGINS
4   CHECKVAR
5   CHECKWKSSELECTION
6   CLEARWORKSHEET
7   CLR
8   DBCPAGE
```

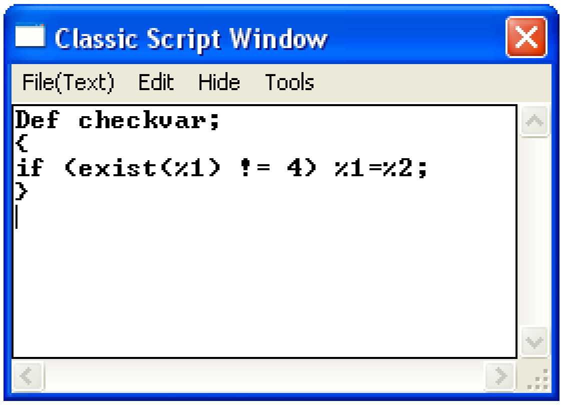
To see the definition of any system macro:

3. To see how a system macro is defined, type **def *macroname***. For example, type the following in the Script Window:

```
def checkvar
```

4. Press **ENTER**.

Origin responds to the Script Window as:



```
Def checkvar;
{
if (exist(%1) != 4) %1=%2;
}
```

The %1 notation in the macro definition indicates that this macro takes one argument.

To define a new macro as a system macro:

5. Type the following in the Script Window:

```
def graph {
    set %1 -s 1;layer -i %1
}
```

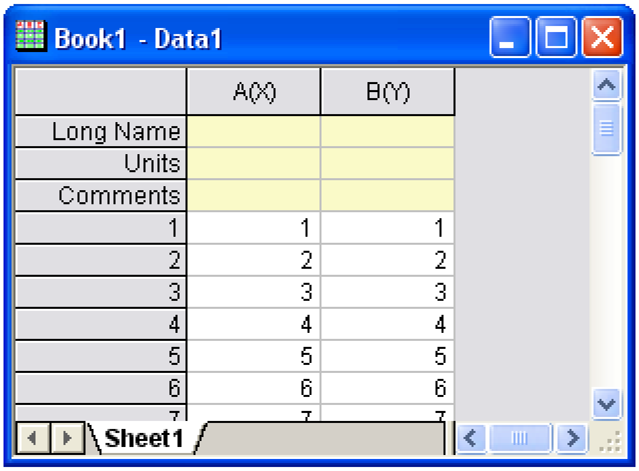
6. Press **ENTER**.

Notes: We used the set data set ?s 1 command to show a data plot. The layer -i data set command adds (plots) the named data set onto the active layer.

To call a macro:

7. Click the **New Worksheet** button

8. Create a worksheet named *Book1* and type in the following data:



	A(X)	B(Y)
Long Name		
Units		
Comments		
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7

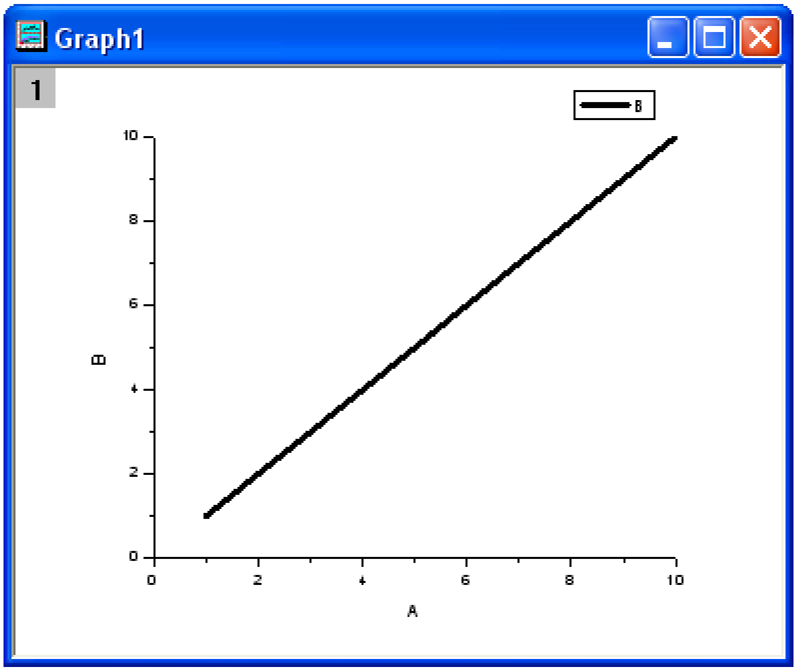
9. Click the **New Graph** button

10. In the Script Window, type the following:

```
graph book1_b
```

11. Press **ENTER**.

A line plot of the data set book1_b is included in the graph window.



We will modify the macro definition so that it creates a scatter plot using a red, ?up triangle? as the symbol.

You could hard code the appropriate values for scatter plot, red, and up triangle in the macro definition, but it is more efficient to pass the value of a variable as an argument. This way, the macro may be used in other instances when you want to set the color and symbol shape to something other than red and upward-pointing triangle .

> **Notes:** LabTalk often uses integer values to specify plot details. If you look at the color palette, for instance (from the menu, **Format: Color Palette**), you will see (assuming that you have not modified the default color palette) that black =1, red = 2, green = 3, blue = 4, etc. For more information, see documentation on the **Set** command in the LabTalk Language Reference section of the Programming Help file.

12. Type the following in the Script Window to redefine the graph macro:

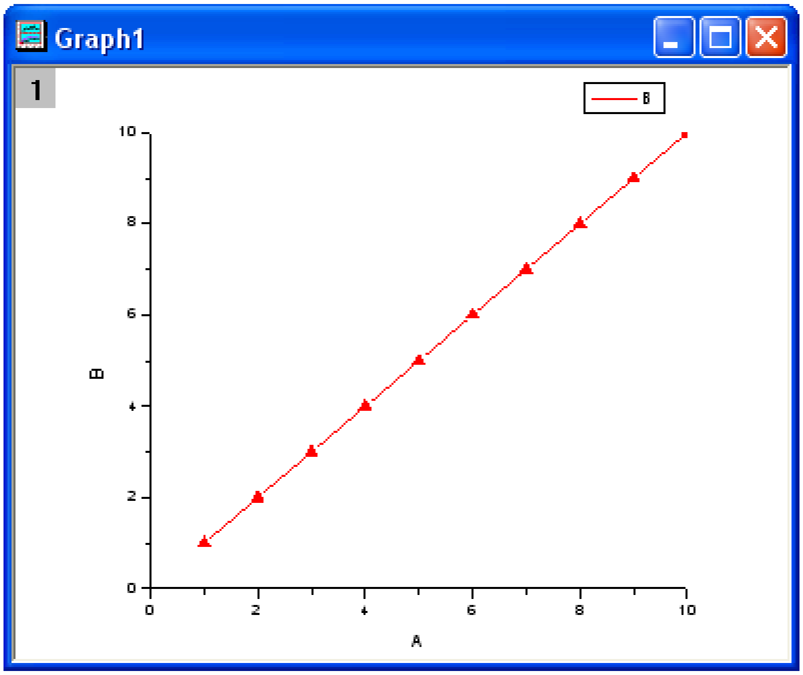
```
def graph {
    set %1 -s 1;
    layer -i %1;
    set %1 -c %2;
    set %1 -k %3;
}
```

13. Type the following:

```
graph book1_b 2 3
```

14. Press **ENTER**.

This plots the data set book1_b as red, upward-pointing triangles.



To better understand what we actually did, let?s examine our macro line by line.

First, we used the **def** command to tell Origin that we are defining a macro.

```
def graph
```

Secondly, we used the LabTalk **set** command with the **? s** option.

The **set ?s** command syntax is:

```
set dataset -s value
```

where *dataset* is the name of a data set, and *value* is either 1 (show plot) or 0 (hide plot),

```
set %1 ?s 1;
```

The layer **?i#** command syntax is:

```
layer ?igraphType dataset;
```

Note that the data set name has been assigned to %1.

The **set ?c** command syntax is:

```
set ?c color#
```

This is used to specify the plot symbol color. Note that color will be assigned to %2.

The **set ?k** command syntax is:

```
set ?k shape#
```

This is used to specify symbol shape. Note that symbol shape is assigned to %3.

When we execute our macro by typing

```
graph book1_b 2 3
```

we are passing three arguments to the macro:

- book1_b, which is substituted for %1.
- 2, which is substituted for %2.
- 3, which is substituted for %3.

Note that it is merely coincidence that we chose to substitute a value of 2 for %2, and a value of 3 for %3. We could have chosen any allowed value for symbol color or shape.

Remember that any macro that you define is only available for the duration of your Origin session. If you restart Origin, you cannot execute your macro until you define it again. If you want your macro to be defined automatically when you start Origin, you can save your macro definition to Origin's MACROS.CNF file. Each time Origin starts, it reads MACROS.CNF, and your macro is defined.

> **Notes:** MACROS.CNF is located in the Origin software folder. Because of a turf battle over the .CNF file extension, it will probably only be listed as MACROS and will display a terminal icon. In reality, this file is a text file and can be opened in any text editor, such as Notepad and Origin Code Builder.

For more information on macros, see **Help: Programming: Labtalk** in the Help menu.

11.7.7 Loading and Compiling your Origin C Function using LabTalk script

Before an Origin C function can be used, it must be compiled and linked in the current Origin session. Origin provides the following method to programmatically compile and link a source file, or to programmatically build a workspace, from LabTalk.

```
err = run.LoadOC("myFile",[option]);
```

> **Notes:** For more on the LabTalk run object in **Help: Programming:Labtalk** Help.

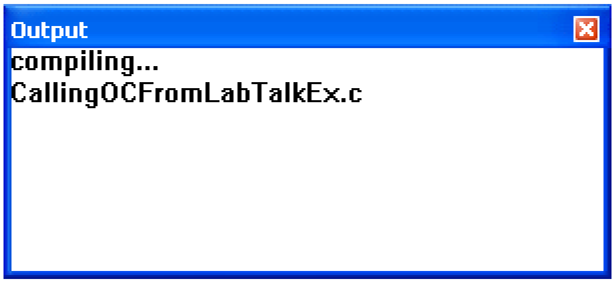
The following example demonstrates how to programmatically load and compile an Origin C source file.

To begin this tutorial:

1. On the **Standard** Toolbar, click the **Code Builder** button .
2. Return to the Origin workspace and open a New Project (**File: New?Project**).
3. Open the Script Window (**Window: Script Window**) and type the following:

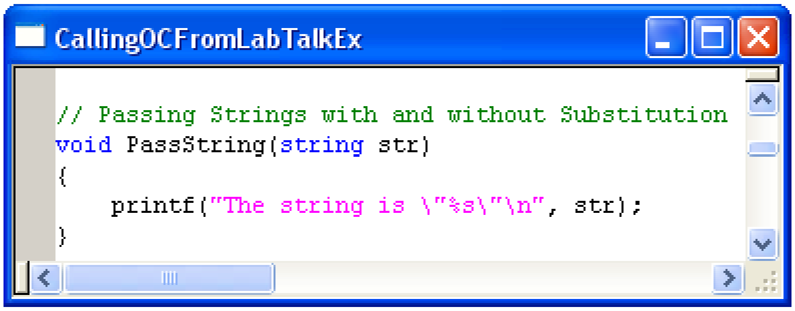
```
string fld$="Samples\Origin C Examples\Programming Guide\Calling Functions\";
string fname$=System.path.program$ + fld$ + "CallingOCFromLabTalkEx.c";
run.LoadOC(%(fname$), 1);
```

The CallingOCFromLabTalkEx.c function is compiled now and can be found under the User folder in Code Builder's Workspace view. The Output Window reports as follows in the Code Builder workspace:



```
compiling...
CallingOCFromLabTalkEx.c
```

The Origin C functions in CallingOCFromLabTalkEx.c are now accessible. You can call the following section from the C file. Note the comments (the green text with leading //).



```
// Passing Strings with and without Substitution
void PassString(string str)
{
    printf("The string is \"%s\"\n", str);
}
```

To execute the PassString function:

4. Return to the Origin workspace, open the Script Window (**Window: Script Window**) and type in the following:

```
PassString abc
```

5. Press **ENTER**. Origin returns:

```
The string is "abc"
```

11.7.8 Using .OGS Files to Store Script

As an alternative to associating your LabTalk script or Origin C function with a button, you could save your script to .OGS files. The advantage is that these .OGS files are self-contained and can be called from many buttons.

These .OGS files are organized by sections. Sections are identified by a name surrounded by square brackets, as in this example:

```
[Main]
```

To execute the code in a portion of a .OGS file, you need only identify the .OGS file and refer to the section containing the code by name, as in this example:

```
run.section(test.ogs, Main)
```

Most of Origin's menu and toolbar commands run LabTalk script in a .OGS file. These files can be opened and edited in Code Builder.

In this tutorial, we will create a new .OGS file, and associate the .OGS file with a new toolbar button.

Notes: This tutorial assumes that you have already created and saved an Origin C file called test.c, as prescribed under **Tutorial: Organizing and Accessing Origin C Functions**.

To create a new .OGS file:

1. From the **Code Builder** menu, select **File: New**.
2. In the **New File** dialog box, select **LabTalk Script File**.
3. In the **File Name** text box, type:

```
Test
```

4. Click **OK**.

You now have an empty document called **test.ogs**.

We will use the run.LoadOC script command to programmatically compile and link the test.C Origin C source file. The advantage of this method is that it allows you to program your buttons or other user-created visual objects to make behind-the-scenes calls to your Origin C functions.

To make a call to an ?uncompiled? Origin C function from Origin:

5. In the blank **test.ogs** window, type the following:

```
[CreateGraph]
run.LoadOC("test.c");
Plot_Data("scatter","book1_b");
```

6. From the Code Builder menu, select **File: Save As** and save the file to your main Origin software folder.

7. Return to the Origin workspace.

8. From the Origin menu, select **View: Toolbars**. This opens the **Customize Toolbar** dialog.

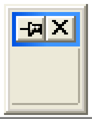
9. On the **Toolbars** tab, click the **New** button to open the **New Toolbar** dialog.

10. Type in the following name for your new toolbar:

```
My Toolbar
```

11. Click **OK**.

My Toolbar is added to the **Toolbars** list. A new toolbar is added to the Origin workspace.

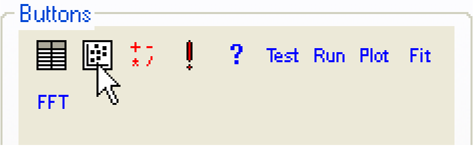


Notes: The toolbar may be a bit hard to spot because it does not contain any buttons.

12. Return to the **Customize Toolbars** dialog and select the **Button Groups** tab.

13. Scroll to the bottom of the **Groups** list and select **User Defined**.

14. Select the second button in this group.



15. Click the **Settings** button. This opens the **Button Settings** dialog box

16. In the File Name text box, type **test.ogs**.

17. In the Section Name text box, type **CreateGraph**. We are not passing arguments to the section, so we can skip the **Argument List** text box.

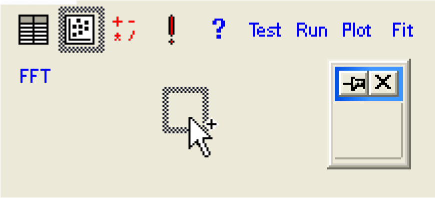
18. In the **Tool Tip Text** box, type **CreateGraph**. A Tool Tip is the message that displays when you mouse over a toolbar button.

19. In the **Status Bar** text box, type **Example, plotting data from Origin C** as Status Bar Text. When you mouse over a toolbar button, the Status Bar message displays in the lower left corner of your Origin workspace.

20. In the **Context** group, verify that the **Windows** radio button is selected and clear the **Graph**, **Matrix**, **Layout**, and **Excel** check boxes. Leave only **Worksheet** selected. This limits toolbar availability to active worksheets.

21. Click **OK**.

22. Point to the toolbar button, hold down the left mouse button and drag the button to the floating toolbar.



23. Click **Close** to close the **Customize Toolbar** dialog.

To test this method, close and restart Origin. Remember that the second argument to our function is book1_b, so the Origin workspace will need to have a worksheet named book1, a B(Y) column and some data in both the X and Y columns. Note, too, that our toolbar button is grayed out when a graph is the active window.

This concludes the tutorial on **Advanced Scripting Using LabTalk**.

11.8 Introduction to X-Functions

11.8.1 Summary

X-Functions provide a structured programming environment that offers a framework for building Origin tools. Different from the simple GetN box, creating tools by using X-Functions allows the user to focus on the actual data processing code and not have to worry about codes for the user interface.

Most of the dialogs/functions in Origin 8 are X-Functions, and many of them can be run from both menu and command line mode. The flexibility of running X-Functions makes them an attractive approach to customizing Origin

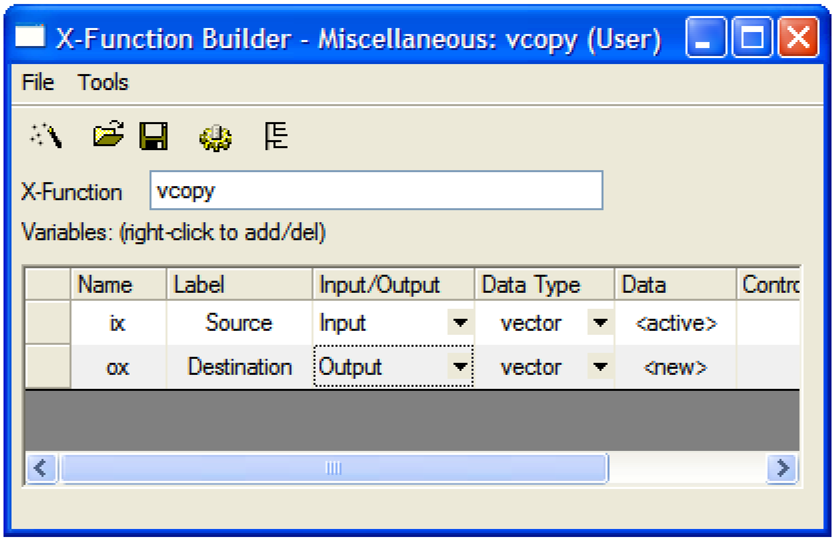
Minimum Origin Version Required: Origin 8.0 SR0

11.8.2 What you will learn

- How to create an X-Function
- How to make the X-Function script accessible
- How to use the X-Funciton in dialog mode

11.8.3 Create an X-Function

1. Select **Tools: X-Function Builder** or press **F10** to open the **X-Function Builder** dialog
2. Set **Name**, **Label** and **Data** of the 1st variable as **ix**, **Source** and **<active>**
3. Right click in the list panel and select **Add Variables** from the context menu.
4. Set **Name**, **Label**, **Input/Output** and **Data** of the 2nd variable as **ox**, **Destination**, **Output** and **<new>**
5. Select **File:Save** to save the x-function as vcopy



X-Function: vcopy

Variables: (right-click to add/del)

	Name	Label	Input/Output	Data Type	Data	Contro
	ix	Source	Input	vector	<active>	
	ox	Destination	Output	vector	<new>	

6. Click to open **Code Builder**
7. Add the following codes in the vcopy function in code builder

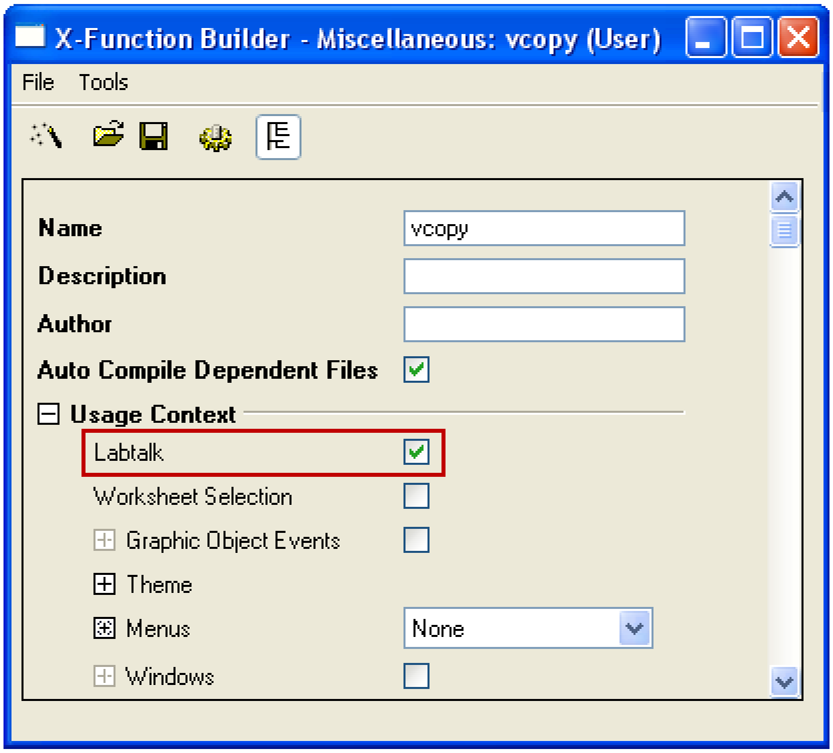
```
void vcopy(const vector& ix, vector& ox)
{

        if (!ix || !ox)
            XF_THROW(CER_NO_DATA);

        ox = ix;

}
```

11.8.4 Making the X-Function Script Accessible

1. Click the **Return to Dialog** button in **Code Builder**
2. In the **X-Function Builder**, save your changes

3. Open the X-Function in **Tree View** by clicking

4. Open the **Usage Context** branch. Make sure the **Labtalk** check box is selected



Name: vcopy

Description:

Author:

Auto Compile Dependent Files ☑

Usage Context

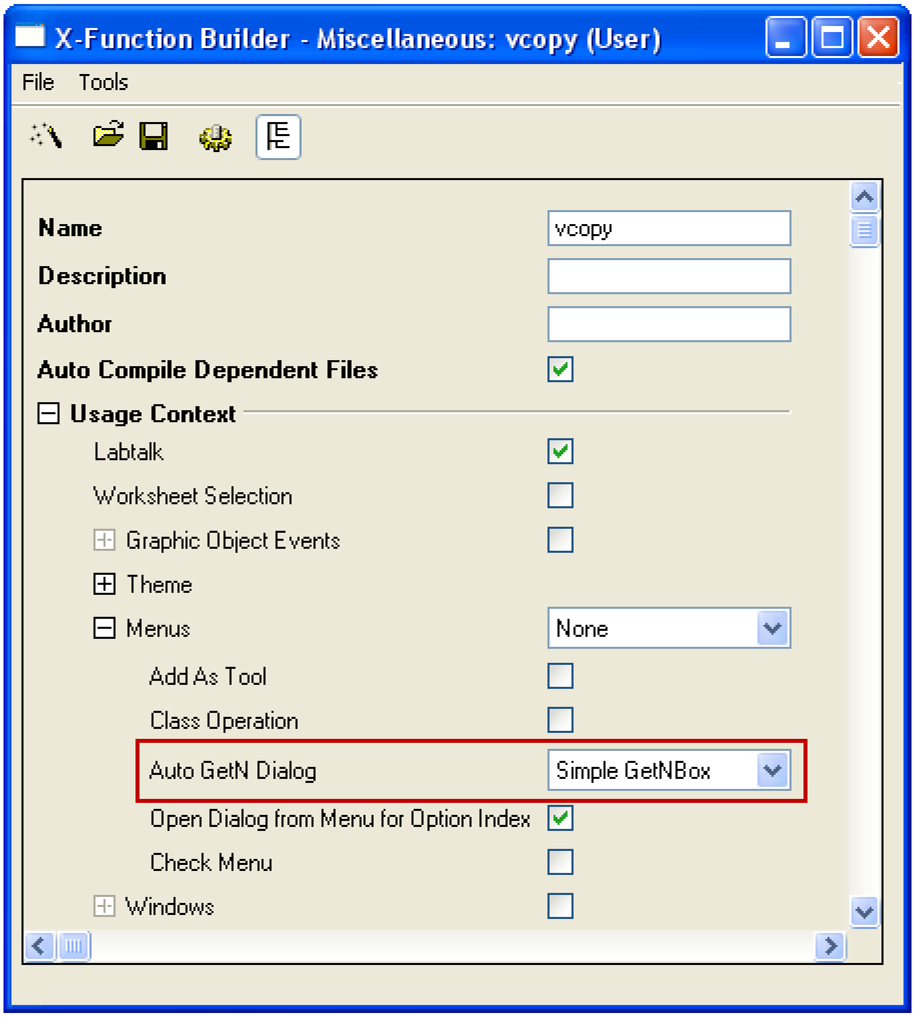
- Labtalk ☑
- Worksheet Selection □
- Graphic Object Events □
- Theme
- Menus: None
- Windows □

5. Save the x-function and close the **X-Function Dialog**
6. Fill column(A) with row numbersw in the active worksheet (Highlight column(A), right-click and select **Fill Column with: Row Numbers**)
7. Type the following script in the command window, Column(A) will be copied to Column(B)

```
vcopy col(a) col(b)
```

11.8.5 Using the X-Function in Dialog Mode

1. Open the **X-Function Dialog** and open VCOPY.OXF in **Tree View**
2. Open the **Usage Context** branch
3. Open the **Menus** branch, make sure **Simple GetNBox** is selected from the **Auto GetN Dialog** list box



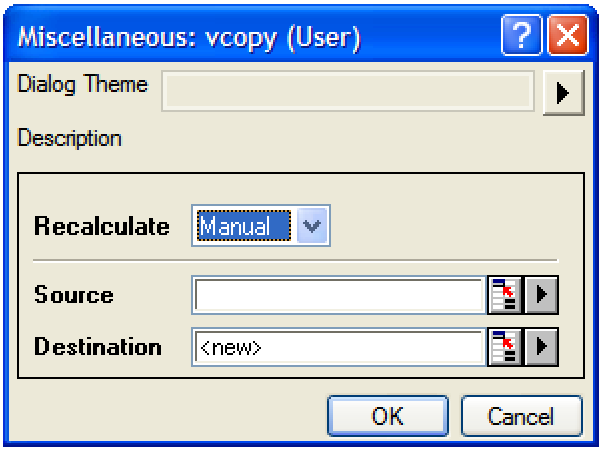
X-Function Builder - Miscellaneous: vcopy (User)

- Name: vcopy
- Description:
- Author:
- Auto Compile Dependent Files ☑
- ⊟ Usage Context
  - Labtalk ☑
  - Worksheet Selection □
  - ⊞ Graphic Object Events □
  - ⊞ Theme
  - ⊟ Menus: None
    - Add As Tool □
    - Class Operation □
    - Auto GetN Dialog: Simple GetNBox
    - Open Dialog from Menu for Option Index ☑
    - Check Menu □
  - ⊞ Windows □

4. Save the x-function and close the **X-Function Dialog**

5. Type following script in the command window, Dialog of VCOPY.OXF will be opened

```
vcopy -d
```



Miscellaneous: vcopy (User)

- Dialog Theme:
- Description
- Recalculate: Manual
- Source:
- Destination: <new>

11.9 How to Create a Wizard

11.9.1 Summary

A wizard is a graphical user interface that includes a series of dialogs to direct a user to complete a mission step by step. A wizard makes a complex task easier to perform. Origin provides several classes in Origin C for users to develop a wizard. The dialog for each step in the wizard can be created using an X-Function.

In this example, the wizard will perform a normality test and then a one-sample t-test for data in a column. The normality test's result can be shared in one-sample t-test.

> **Note:** This tutorial requires the Develop Kit.

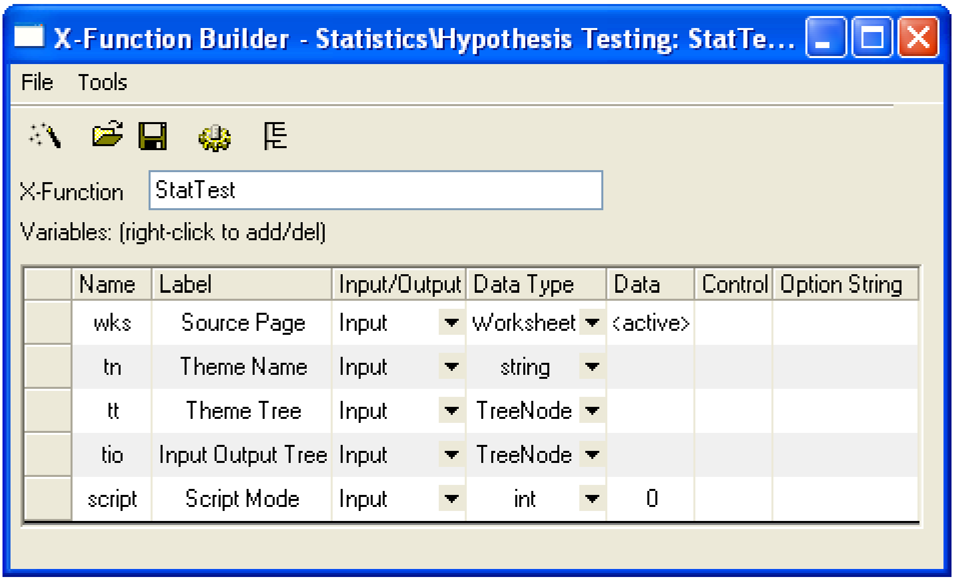
Minimum Origin Version Required: Origin 8.1SR0

11.9.2 What you will learn

- How to create an X-Function.
- How to share a variable or a DataRange in different steps.
- How to call an X-Function in OriginC.
- How to create a wizard.

11.9.3 Create four X-Functions

1. Select **Tools: X-Function Builder** or press **F10** to open **X-Function Builder dialog**
2. Add the variables as follows and save the X-Function as "StatTest" in the User Files folder, User Files\X-Functions\Statistics\Hypothesis Testing.

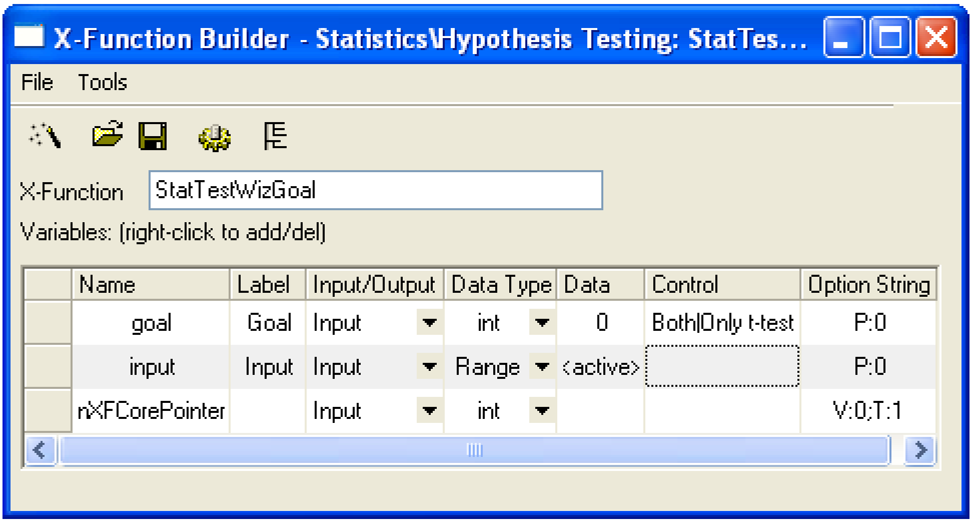

X-Function Builder - Statistics\Hypothesis Testing: StatTe...
File
Tools

X-Function StatTest

Variables: (right-click to add/del)

	Name	Label	Input/Output	Data Type	Data	Control	Option String
	wks	Source Page	Input	Worksheet	<active>		
	tn	Theme Name	Input	string			
	tt	Theme Tree	Input	TreeNode			
	tio	Input Output Tree	Input	TreeNode			
	script	Script Mode	Input	int	0		

3. Click the **New X-Function Wizard** button. Add the variables as follows and save the X-Function as "StatTestWizGoal" in the User Files folder, User Files\X-Functions\Statistics\Hypothesis Testing.

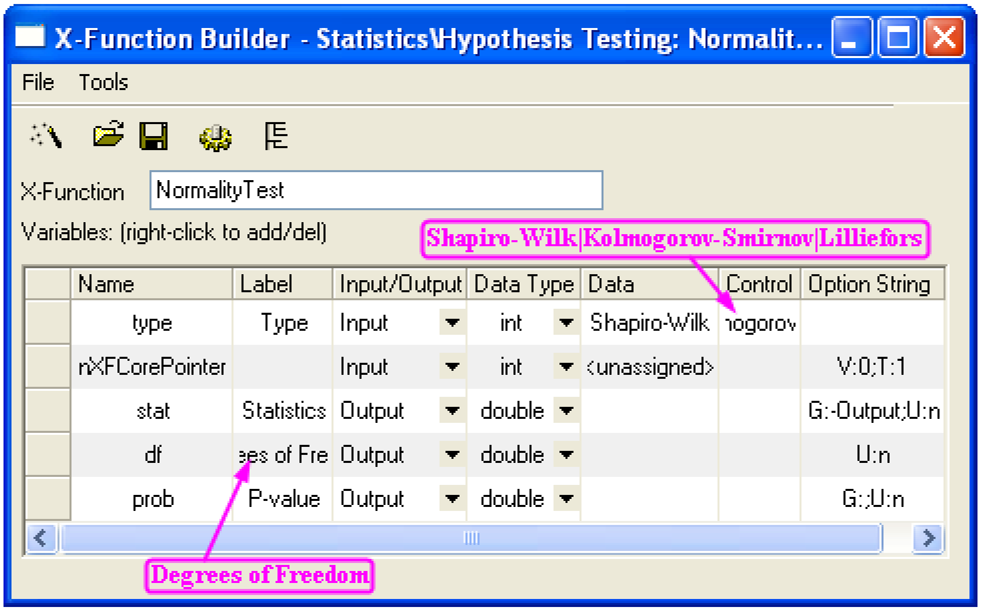

X-Function Builder - Statistics\Hypothesis Testing: StatTes...
File
Tools

X-Function StatTestWizGoal

Variables: (right-click to add/del)

Name	Label	Input/Output	Data Type	Data	Control	Option String
goal	Goal	Input	int	0	Both|Only t-test	P:0
input	Input	Input	Range	<active>		P:0
nXFCorePointer		Input	int			V:0;T:1

4. Click the **New X-Function Wizard** button. Add the variables as follows and save the X-Function as "NormalityTest" in the User Files folder, User Files\X-Functions\Statistics\Hypothesis Testing.


X-Function Builder - Statistics\Hypothesis Testing: Normalit...
File
Tools

X-Function NormalityTest

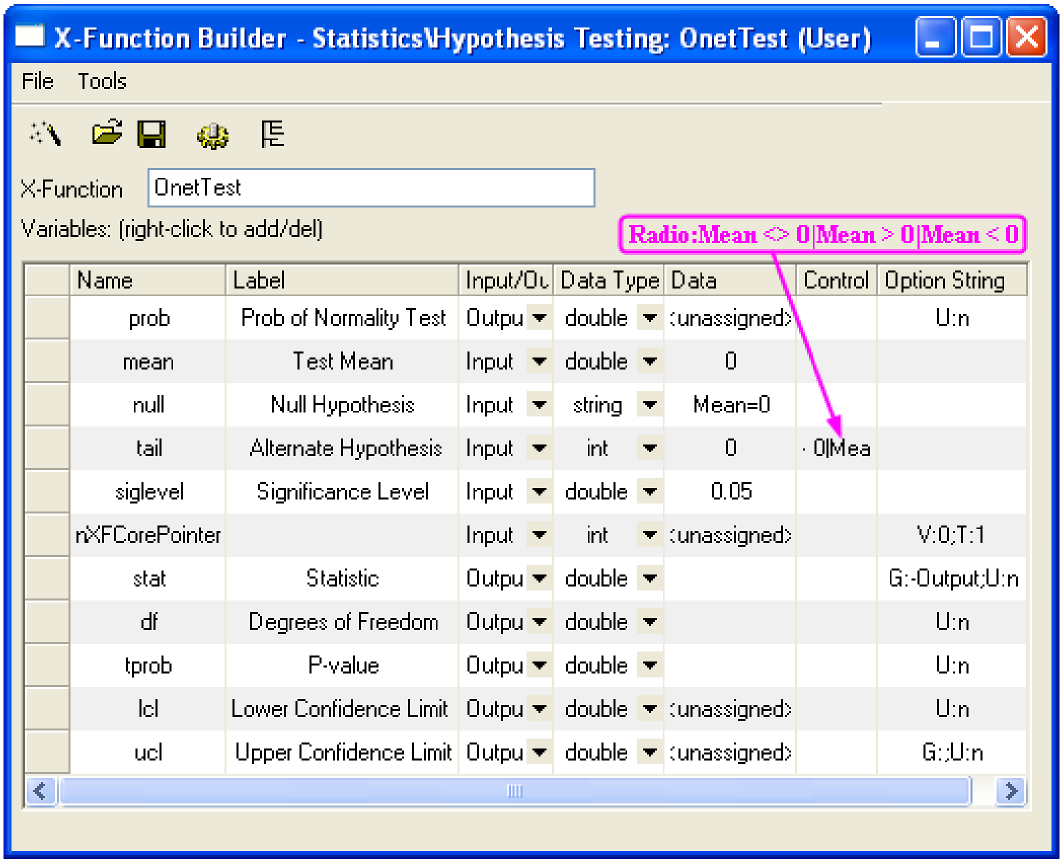
Variables: (right-click to add/del)

Shapiro-Wilk|Kolmogorov-Smirnov|Lilliefors

Name	Label	Input/Output	Data Type	Data	Control	Option String
type	Type	Input	int	Shapiro-Wilk	nogorov	
nXFCorePointer		Input	int	<unassigned>		V:0;T:1
stat	Statistics	Output	double			G:-Output;U:n
df	ees of Fre	Output	double			U:n
prob	P-value	Output	double			G:;U:n

Degrees of Freedom

5. Click the **New X-Function Wizard** button. Add the variables as follows and save the X-Function as "OnetTest" in the User Files folder, User Files\X-Functions\Statistics\Hypothesis Testing.



X-Function Builder - Statistics\Hypothesis Testing: OnetTest (User)
File
Tools

X-Function: OnetTest

Variables: (right-click to add/del)

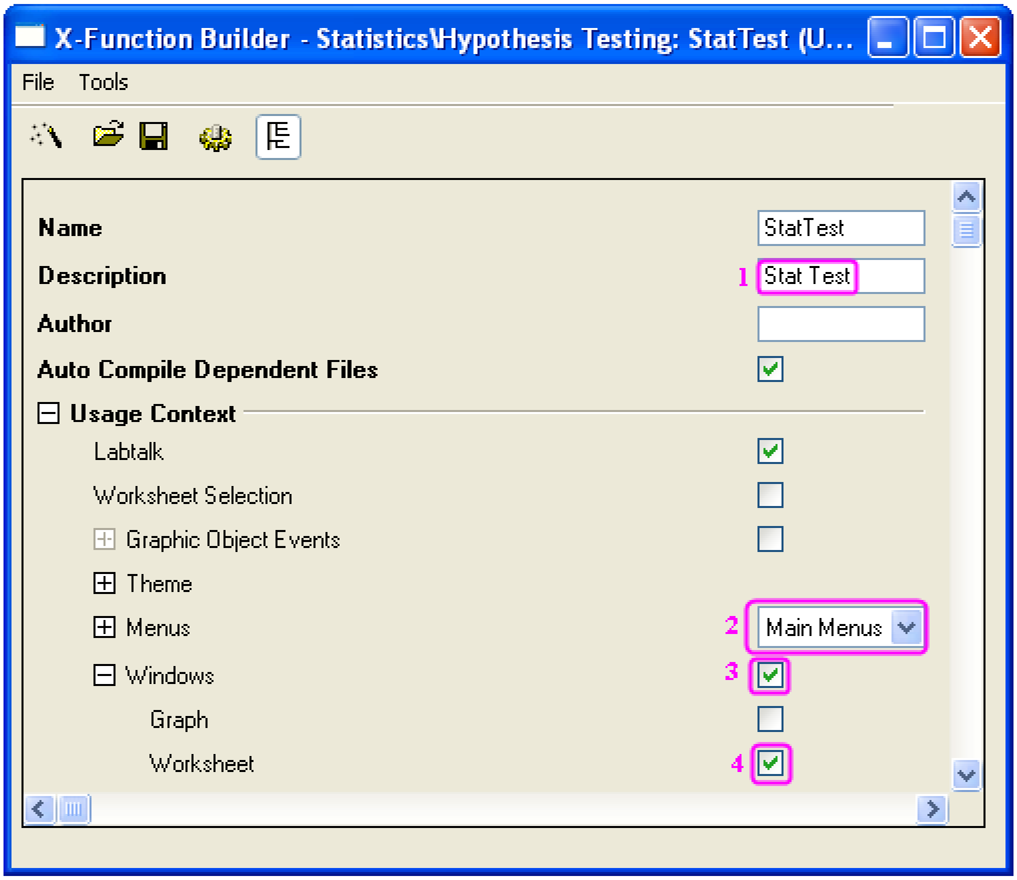
Radio:Mean <> 0|Mean > 0|Mean < 0

Name	Label	Input/Ou	Data Type	Data	Control	Option String
prob	Prob of Normality Test	Outpu	double	<unassigned>		U:n
mean	Test Mean	Input	double	0		
null	Null Hypothesis	Input	string	Mean=0		
tail	Alternate Hypothesis	Input	int	0	· 0|Mea	
siglevel	Significance Level	Input	double	0.05		
nXFCorePointer		Input	int	<unassigned>		V:0;T:1
stat	Statistic	Outpu	double			G:-Output;U:n
df	Degrees of Freedom	Outpu	double			U:n
tprob	P-value	Outpu	double			U:n
lcl	Lower Confidence Limit	Outpu	double	<unassigned>		U:n
ucl	Upper Confidence Limit	Outpu	double	<unassigned>		G:;U:n

Note that the X-Functions **NormalityTest** and **OnetTest** have the same variable "prob", which is a shared variable and will be declared in the source file.

11.9.4 Update X-Function's Property in Tree View

1. Open the X-Function **StatTest**. Click the **TreeView** button to open the Tree View. Make the following settings in the Tree View.



X-Function Builder - Statistics\Hypothesis Testing: StatTest (U...
File Tools

Name: StatTest
Description: Stat Test
Author:
Auto Compile Dependent Files ☑
⊟ Usage Context
Labtalk ☑
Worksheet Selection □
⊞ Graphic Object Events □
⊞ Theme
⊞ Menus: Main Menus
⊟ Windows ☑
Graph □
Worksheet ☑

2. Click the **Save OXF file** button to save the X-Function.
3. Open the X-Function **StatTestWizGoal**, **NormalityTest** and **OnetTest** respectively in X-Function Builder. Click the **TreeView** button and type "Select Wizard Goal", "Normality Test" and "One-Sample t-test" in the **Description** edit box of each X-Function's Tree View, which will be shown in the dialogs.

11.9.5 Create Files for the Wizard

- Click the **Code Builder** button on the **Standard** toolbar. In Code Builder, click the **New** button. In the **New File** dialog, select **H File**, click the **Browse** button, and select the User Files folder, User Files\OriginC as the new header file's **Location**. Then type StatTestWiz in the **File Name** edit box. Click **OK** to close the dialog.

Add the following script to StatTestWiz.h file.

```
#ifndef __STAT_TEST_WIZ_H__
#define __STAT_TEST_WIZ_H__
```

```
#include <..\OriginLab\XFWiz.h>
#include <..\OriginLab\XFCore.h>
#include <..\OriginLab\XFWizard_utils.h>

class StatTestWizCore : public XFCore
{
public:
   StatTestWizCore();

public:
   void ChangeGoal(int nGoal);
   DataRange GetRange();

   int nStep;
protected:

};

int stat_test_run_wiz_nodlg(LPCSTR lpcszThemeName = NULL, const XFWizTheme *pXFWizTheme
= NULL, const XFWizInputOutputRange *pXFWizIO = NULL, DWORD dwOPUID = 0);

int stat_test_open_wiz_dlg(LPCSTR lpcszThemeName = NULL, const XFWizTheme *pXFWizTheme
= NULL, const XFWizInputOutputRange *pXFWizIO = NULL, DWORD dwOPUID = 0);

#endif  //__STAT_TEST_WIZ_H__
```

Click the **Save** button to save StatTestWiz.h file.

- Repeat the same operation to create a new **C File**, StatTestWiz.c .

Add the following script to StatTestWiz.c file.

```
///////////////////////////////////////////////////////////////////////////////////
#include <..\OriginLab\XFWizManager.h>

#include <..\OriginLab\WizOperation.h>
#include <..\OriginLab\XFWizNavigation.h>

#include <..\OriginLab\XFWizScript.h>
#include <..\OriginLab\XFWizDlg.h>

///////////////////////////////////////////////////////////////////////////////////
// Include your own header files here.
#include "StatTestWiz.h"

enum
{
       GOAL_ALL = 0,
       GOAL_SIMPLE,
};

//Names of three X-Functions
#define STR_STEP_GOAL  "StatTestWizGoal"
#define STR_STEP_Normal       "NormalityTest"
#define STR_STEP_TTest "OnetTest"

//Names of steps shown in the wizard.
#define STR_LABEL_STEP_GOAL   "Goal"
#define STR_LABEL_STEP_Normal "Normality Test"
#define STR_LABEL_STEP_TTest  "One-Sample t-test"

///////////////////////////////////////////////////////////////////////////////////
//Class StatTestWizTheme
class StatTestWizTheme : public XFWizTheme
{
public:
   StatTestWizTheme();
};

//Name of the variable prob shared by X-Functions NormalityTest and OnetTest
```

```
#define STR_GETN_VAR_SHARED_NProb "prob"

StatTestWizTheme::StatTestWizTheme()
:XFWizTheme()
{
   m_saSharedList.Add(STR_GETN_VAR_SHARED_NProb); //Add the shared variable
}

///////////////////////////////////////////////////////////////////////////////////
class StatTestWizInputOutputRange : public XFWizInputOutputRange
{
};

///////////////////////////////////////////////////////////////////////////////////
//Class StatTestWizManager

#define STR_CLASS_NAME_TEST   "StatTestWiz"
#define TEST_VERSION_NUMBER   1.0

class StatTestWizManager : public XFWizManager
{
public:
   StatTestWizManager(LPCSTR lpcszThemeName = NULL, const XFWizTheme *pXFWizTheme
= NULL, const XFWizInputOutputRange *pXFWizIO = NULL, DWORD dwUIDOp = 0);

protected:
   virtual double GetVersion() { return TEST_VERSION_NUMBER; }
   virtual XFCore* CreateXFCore() { return new StatTestWizCore; }
   virtual XFWizTheme* CreateXFWizTheme() { return new StatTestWizTheme; }
   virtual XFWizInputOutputRange* CreateXFWizInputOutputRange()
             { return new StatTestWizInputOutputRange; }
   virtual string GetClassName() { return STR_CLASS_NAME_TEST; }
};

StatTestWizManager::StatTestWizManager(LPCSTR lpcszThemeName, const XFWizTheme
*pXFWizTheme, const XFWizInputOutputRange *pXFWizIO, DWORD dwUIDOp)
: XFWizManager(lpcszThemeName, pXFWizTheme, pXFWizIO, dwUIDOp)
{
   StringArray saMapXFNames = {STR_STEP_GOAL, STR_STEP_Normal, STR_STEP_TTest};
   StringArray saMapXFLabels = {STR_LABEL_STEP_GOAL, STR_LABEL_STEP_Normal,
                                 STR_LABEL_STEP_TTest};
   m_saMapXFNames = saMapXFNames;
   m_saMapXFLabels = saMapXFLabels;
   ASSERT( m_saMapXFNames.GetSize() == m_saMapXFLabels.GetSize() );

   StringArray saDefaultXFNames = {STR_STEP_GOAL, STR_STEP_Normal, STR_STEP_TTest};
   m_saDefaultXFNames = saDefaultXFNames;

   m_strRunDlgName = _L("Stat Test");
}

///////////////////////////////////////////////////////////////////////////////////
//Class StatTestWizCore

StatTestWizCore::StatTestWizCore()
:XFCore()
{
   StringArray vsXFsRecalculateShown = {STR_STEP_GOAL};
   m_vsXFsRecalculateShown = vsXFsRecalculateShown;
   nStep = GOAL_ALL;
}

//Select steps in the Goal Step
void StatTestWizCore::ChangeGoal(int nGoal)
{
   XFWizNavigation *pXFWizNavg = (XFWizNavigation *)GetXFWizNavigation();
   ASSERT(pXFWizNavg);

   nStep = nGoal;

   if ( pXFWizNavg )
   {
      StringArray saXFNames;
      saXFNames.Add(STR_STEP_GOAL);
      switch (nGoal)
      {
      case GOAL_ALL:
         saXFNames.Add(STR_STEP_Normal);
         saXFNames.Add(STR_STEP_TTest);
         break;
      case GOAL_SIMPLE:
         saXFNames.Add(STR_STEP_TTest);
         break;
      }

      pXFWizNavg->SetSteps(saXFNames);

   }
}

//Get input DataRange in the Goal Step.
DataRange StatTestWizCore::GetRange()
{
   XFWizNavigation *pXFWizNavg = (XFWizNavigation*)GetXFWizNavigation();
   XFWizInputOutputRange* pIORange = pXFWizNavg->GetXFWizInputOutputRange();

   DataRange drInput;
   if(!pIORange)
   {
      error_report("Fail to get io ranges!");
      return drInput;
   }

   Array<DataRange&> drs;
   //Get input DataRange.
   if(!pIORange->Get(&drs, STR_STEP_GOAL, true))
   {
      error_report("Fail to get range from WizCore!");
      return drInput;
   }

   drInput = drs.GetAt(0);

   return drInput;
}

////////////////////////////////////////////////////////////////////////////////////

int stat_test_run_wiz_nodlg(LPCSTR lpcszThemeName, const XFWizTheme *pXFWizTheme, const
XFWizInputOutputRange *pXFWizIO, DWORD dwOPUID)
{
   TEMPLATE_run_wiz_nodlg(StatTestWizManager, lpcszThemeName, pXFWizTheme, pXFWizIO,
dwOPUID)
}

int stat_test_open_wiz_dlg(LPCSTR lpcszThemeName, const XFWizTheme *pXFWizTheme, const
XFWizInputOutputRange *pXFWizIO, DWORD dwOPUID)
{
   TEMPLATE_open_wiz_dlg(StatTestWizManager, lpcszThemeName, pXFWizTheme, pXFWizIO,
dwOPUID)
}

int stat_test_run_wiz(UINT msg, const XFWizTheme *pXFWizTheme, const
XFWizInputOutputRange *pXFWizIO, DWORD dwOPUID, int nExeMode)
{
   TEMPLATE_run_wiz(StatTestWizManager, msg, pXFWizTheme, pXFWizIO, dwOPUID, nExeMode)
}
```

Click the **Save** button to save StatTestWiz.c file.

Note that StatTestWiz.c should be compiled after the X-Function **StatTest** is compiled, since the included files in StatTestWiz.c are not yet in the workspace until the X-Function **StatTest** is compiled. In fact StatTestWiz.h is included in X-Function **StatTest**, so StatTestWiz.c will be compiled automatically when X-Function **StatTest** is compiled.

11.9.6 Add Script for X-Functions

Script for X-Function StatTest

In the **X-Function Builder**, click the **Open** button and open the X-Function StatTest. Click the **Edit X-Function in Code Builder** and add the following script.

- Include header files

```
#include <..\OriginLab\XFWiz.h>
#include <..\OriginLab\WizOperation.h>
#include <..\OriginLab\XFCore.h>
#include <..\OriginLab\XFWizNavigation.h>
#include <..\OriginLab\XFWizManager.h>
#include <..\OriginLab\XFWizScript.h>
#include <..\OriginLab\XFWizDlg.h>

#include <..\OriginLab\XFWizard_utils.h>

#include <..\OriginLab\WksOperation.h>

#include <event_utils.h>

#include "StatTestWiz.h"
```

- StatTest()

Add the function body, which specifies the dialog mode.

```
if( script )
   stat_test_run_wiz_nodlg(tn);
else
   stat_test_open_wiz_dlg(tn);
```

- StatTest_before_execute()

Add the function body, which determines not to show this dialog before the wizard is opened.

```
   nRet = XFEVT_PROCEED_NO_DLG;
```

Click **Compile** button to compile the file. Then click **Return to Dialog** button to return to **X-Function Builder**. In the **X-Function Builder**, click **Save OXF file** button to save the X-Function.

Script for X-Function StatTestWizGoal

Open the X-Function StatTestWizGoal. Click **Edit X-Function in Code Builder** button, add the following script.

- Include header files

```
#include "StatTestWiz.h"
```

- Add a static function _check_input()

This function is used to check whether the input DataRange is a single column.

```
static bool _check_input(const TreeNode trGetN, string& strErr)
{
   TreeNode trRange = trGetN.input;
   DataRange drInput;
   drInput.Create(trRange.strVal);

   if( drInput.GetNumRanges() == 0 )
   {
      strErr = "Input can't be empty, and it should be a valid column.";
      return false;
   }
   else
   {
      if( drInput.GetNumRanges() == 1)
      {
         Worksheet wksInput;
         int nC1, nC2;
         drInput.GetRange(wksInput, nC1, nC2);
         if( nC1 == nC2 )
            return true;
      }

          strErr = "Please select one column.";
          return false;
   }
}
```

- StatTestWizGoal_event1()

Add the function body, which updates the dialog.

```
   StatTestWizCore* pstatwc = (StatTestWizCore*)get_xf_core_handler(trGetN);
   ASSERT(pstatwc);

   //Update the Wizard page.
   if ( 0 == lstrcmp(lpcszNodeName, "goal") )
      pstatwc->ChangeGoal(trGetN.goal.nVal);


   //Error message is shown at the bottom of the dialog,
   //and OK button is disenabled for incorrect choice of DataRange.
   bOKEnable = _check_input(trGetN, strErrMsg);

   return false;
```

Click **Compile** button to compile the file. Then click **Return to Dialog** button to return to **X-Function Builder**, and click **Save OXF file** button to save the X-Function.

Script for X-Function NormalityTest

Open the X-Function **NormalityTest**. Click the **Edit X-Function in Code Builder** button and add the following script.

- Include header files

```
#include "StatTestWiz.h"

#include <XFbase.h>
```

- Add a static function _update_GUI()

This function is used to update the dialog's edit boxes for normality test result.

```
static void _update_GUI(TreeNode& trGetN)
{
   vector vRes;
   vRes = _norm_test(trGetN.nXFCorePointer.nVal, trGetN.type.nVal);

   trGetN.stat.dVal = vRes[0];
   trGetN.df.dVal = vRes[1];
   trGetN.prob.dVal = vRes[2];
}
```

- Add a static function _update_strErr()

This function is used to update the string shown at the bottom of the dialog.

```
static void _update_strErr(const TreeNode tr, string& strErr)
{
   if(tr.prob.dVal >= 0.05 && tr.prob.dVal <= 1)
      strErr = "At the 0.05 level, the data was significantly drawn from a
            normally distributed population.";
   else if(tr.prob.dVal < 0.05 && tr.prob.dVal >= 0)
      strErr = "At the 0.05 level, the data was not significantly drawn from a
            normally distributed population.";
   else
      strErr = "There is not enough information to draw a conclusion.";
}
```

Note that the string is divided into two lines shown in the page. It should be a command of one line in the script.

- Add a static function _norm_test()

This function is used to perform Normality Test using related X-Functions.

```
static vector _norm_test(const int nXFCorePointer, const int nType)
{
   StatTestWizCore* pstatwc = (StatTestWizCore*)get_xf_core_handler(nXFCorePointer);
   ASSERT(pstatwc);

   vector vRes(3);
   vRes[2] = -1;
   DataRange drInput;
   drInput = pstatwc->GetRange();
   if( !drInput )
      return vRes;

   vector<string> vsXFName = {"swtest","kstest","lillietest"};
   XFBase xfNorm(vsXFName[nType]);
   if( !xfNorm.SetArg("irng", drInput) )
   {
      error_report("Failed to set argument image type");
      return vRes;
   }
   if( !xfNorm.SetArg("stat", vRes[0]) )
   {
      error_report("Failed to set argument image type");
      return vRes;
   }
   if( !xfNorm.SetArg("df", vRes[1]) )
   {
       error_report("Failed to set argument image type");
       return vRes;
   }
   if( !xfNorm.SetArg("prob", vRes[2]) )
   {
       error_report("Failed to set argument image type");
       return vRes;
   }

   if( !xfNorm.Evaluate() )
   {
      error_report("Failed to evaluate the stats X-Function.");
      return vRes;
   }

   return vRes;
}
```

- NormalityTest()

Update the function body, which exports the result into a worksheet when the **Next** button is pressed.

```
DataRange drInput;
StatTestWizCore* pstatwc = (StatTestWizCore*)get_xf_core_handler(nXFCorePointer);
ASSERT(pstatwc);
drInput = pstatwc->GetRange();

if( !drInput )
   return;
string strBook, strSheet;
if(!drInput.GetBookSheet(strBook, strSheet))
{
   error_report("Workbook and worksheet names can't be obtained.");
   return;
}
WorksheetPage wpData(strBook);

int nLayer = wpData.AddLayer("Normality Test");

if(nLayer >= 0)
{
   Worksheet wksRes = wpData.Layers(nLayer);
   vector<string> vsTypeName = {"Shapiro-Wilk","Kolmogorov-Smirnov","Lilliefors"};
   vector<string> vsNProb = {"Prob<W", "Prob>D", "Prob>D"};
   vector<string> vsParaName = {"Statistic", "DF", ""};
   vsParaName[2] = vsNProb[type];

   vector vRes;
   vRes = _norm_test(nXFCorePointer, type);

   wksRes.Columns(1).SetLongName(vsTypeName[type]);
   for(int ii=0; ii<3; ii++)
   {
      wksRes.SetCell(ii, 0, vsParaName[ii], false);
      wksRes.SetCell(ii, 1, vRes[ii]);
   }
}
else
{
   error_report("New worksheet can't be created.");
}
```

- NormalityTest_event1()

Update the function body, which will update the results in the dialog as the method of normality test changes. Strings shown at the bottom of the dialog will also be updated.

```
_update_GUI(trGetN);
_update_strErr(trGetN, strErrMsg);

return true;
```

- NormalityTest_before_execute()

Update the function body, which will make the edit boxes for results grayed out, and show the result in the dialog.

```
trGetN.stat.Enable = false;
trGetN.df.Enable = false;
trGetN.prob.Enable = false;
```

Click the **Compile** button to compile the file. Then click the **Return to Dialog** button to return to **X-Function Builder**, and click the **Save OXF file** button to save the X-Function.

Script for X-Function OnetTest

Open the X-Function **OnetTest**. Click the **Edit X-Function in Code Builder** button and add the following script.

- Include header files

```
#include "StatTestWiz.h"

#include <XFbase.h>
```

- Define strings

```
const vector<string> vsNull = {"Mean = ","Mean <= ","Mean >= "};
const vector<string> vsAlter = {"Mean <> ","Mean > ","Mean < "};
const vector<string> vsAcceptNull = {"Not significantly different from","Not
significantly greater than","Not significantly less than"};
const vector<string> vsRejectNull = {"significantly different from","significantly
greater than","significantly less than"};
const vector<string> vsProb = {"Prob>|t|", "Prob>t", "Prob<t"};
```

- Add a static function _update_null()

This function is used to update the **Null** edit box.

```
static void _update_null(TreeNode& trGetN, bool bMean = false)
{
   string strNull;

   strNull = vsNull[trGetN.tail.nVal] + ftoa(trGetN.mean.dVal);
   trGetN.null.strVal = strNull;

   if(bMean)
   {
      string strAlter = vsAlter[0] + ftoa(trGetN.mean.dVal) + "|";
      strAlter = strAlter + vsAlter[1] + ftoa(trGetN.mean.dVal) + "|";
      strAlter = strAlter + vsAlter[1] + ftoa(trGetN.mean.dVal);

      trGetN.tail.SetAttribute(STR_COMBO_ATTRIB, strAlter);
   }
}
```

- Add a static function _check_sig_level()

This function is used to check the **Significance Level** edit box value.

```
static bool _check_sig_level(TreeNode& trGetN, string& strErr)
{
   if( trGetN.siglevel.dVal > 0 && trGetN.siglevel.dVal < 1 )
   {
      return true;
   }
   else
   {
      strErr = "Significance Level should be between 0 and 1.";
      return false;
   }
}
```

- Add a static function _update_strErr()

This function is used to define the string for the conclusion of t-test at the bottom based on P-value.

```
static void _update_strErr(const TreeNode tr, string& strErr)
{
   if(tr.tprob.dVal >= tr.siglevel.dVal && tr.tprob.dVal <= 1)
      strErr.Format("Null Hypothesis is %s%s.\r\nAlternative Hypothesis is %s%s.
         At the %s level, the population mean is %s the test mean(%s).",
         vsNull[tr.tail.nVal], ftoa(tr.mean.dVal), vsAlter[tr.tail.nVal],
ftoa(tr.mean.dVal),
         ftoa(tr.siglevel.dVal), vsAcceptNull[tr.tail.nVal], ftoa(tr.mean.dVal) );
   else if(tr.tprob.dVal < tr.siglevel.dVal && tr.tprob.dVal >= 0)
      strErr.Format("Null Hypothesis is %s%s.\r\nAlternative Hypothesis is %s%s.
         At the %s level, the population mean is %s the test mean(%s).",
         vsNull[tr.tail.nVal], ftoa(tr.mean.dVal), vsAlter[tr.tail.nVal],
ftoa(tr.mean.dVal),
         ftoa(tr.siglevel.dVal), vsRejectNull[tr.tail.nVal], ftoa(tr.mean.dVal) );
   else
      strErr = "There is not enough information to draw a conclusion.";
}
```

Note that the command is divided into several lines shown in the page. It should be a command of one line in the script.

- Add a static function _update_GUI()

This function is used to update edit boxes for results in the dialog.

```
static void _update_GUI(TreeNode& trGetN)
{
   vector vRes;
   vRes = _one_sample_t_test(trGetN.nXFCorePointer.nVal, trGetN.mean.dVal,
trGetN.tail.dVal, trGetN.siglevel.dVal);

   trGetN.stat.dVal = vRes[0];
   trGetN.df.dVal = vRes[1];
   trGetN.tprob.dVal = vRes[2];
   trGetN.lcl.dVal = vRes[4];
   trGetN.ucl.dVal = vRes[5];
}
```

- Add a static function _one_sample_t_test()

This function is used to perform One-Sample t-Test using an X-Function.

```
static vector _one_sample_t_test(const int nXFCorePointer, const double dMean, const int
nTail, const double dSiglevel)
{
   DataRange drInput;
   StatTestWizCore* pstatwc = (StatTestWizCore*)get_xf_core_handler(nXFCorePointer);
   ASSERT(pstatwc);

   vector vRes(6);
   vRes[2] = -1;
   drInput = pstatwc->GetRange();
   if( !drInput )
      return vRes;

   vRes[3] = 100 - 100*dSiglevel;

   XFBase xfTTest("ttest1");

   if( !xfTTest.SetArg("irng", drInput) )
   {
      error_report("Failed to set argument irng");
      return vRes;
   }
   if( !xfTTest.SetArg("mean", dMean) )
   {
      error_report("Failed to set argument mean");
      return vRes;
   }
   if( !xfTTest.SetArg("tail", nTail) )
   {
      error_report("Failed to set argument tail");
      return vRes;
   }
   if( !xfTTest.SetArg("alpha", dSiglevel) )
   {
      error_report("Failed to set argument alpha");
      return vRes;
   }

   if( !xfTTest.SetArg("stat", vRes[0]) )
   {
      error_report("Failed to set argument stat");
      return vRes;
   }
   if( !xfTTest.SetArg("df", vRes[1]) )
   {
       error_report("Failed to set argument df");
       return vRes;
   }
   if( !xfTTest.SetArg("prob", vRes[2]) )
   {
       error_report("Failed to set argument prob");
       return vRes;
   }
   if( !xfTTest.SetArg("lcl", vRes[4]) )
   {
       error_report("Failed to set argument lcl");
       return vRes;
   }
   if( !xfTTest.SetArg("ucl", vRes[5]) )
   {
       error_report("Failed to set argument ucl");
       return vRes;
   }

   if( !xfTTest.Evaluate() )
   {
      error_report("Failed to evaluate the ttest1 X-Function.");
      return vRes;
   }

   return vRes;
}
```

- OnetTest()

Update the function body, which exports the result into a worksheet when the **Finish** button is pressed.

```
   DataRange drInput;
   StatTestWizCore* pstatwc = (StatTestWizCore*)get_xf_core_handler(nXFCorePointer);
   ASSERT(pstatwc);

   drInput = pstatwc->GetRange();
   if( !drInput )
      return ;

   string strBook, strSheet;
   if(!drInput.GetBookSheet(strBook, strSheet))
   {
      error_report("Workbook and worksheet names can't be obtained.");
      return;
   }
   WorksheetPage wpData(strBook);

   int nLayer = wpData.AddLayer("One-Sample t-test");

   if(nLayer >= 0)
   {
      Worksheet wksRes = wpData.Layers(nLayer);

      vector<string> vsParaName = {"t Statistic", "DF","", "Conf. Levels in %", "Lower 
Limits", "Lower Limits"};
      vsParaName[2] = vsProb[tail];

      vector vRes;
      vRes = _one_sample_t_test(nXFCorePointer, mean, tail, siglevel);

      wksRes.SetSize(-1, 4);
      wksRes.Columns(0).SetLongName("Test Statistics");
      string strNull = "Null Hypothesis is " + vsNull[tail] + ftoa(mean);
      wksRes.Columns(1).SetLongName(strNull);
      wksRes.Columns(3).SetLongName("Confidence Intervals for Mean");
      for(int ii=0; ii<3; ii++)
      {
         wksRes.SetCell(ii, 0, vsParaName[ii], false);
         wksRes.SetCell(ii, 1, vRes[ii]);

         wksRes.SetCell(ii, 2, vsParaName[ii + 3], false);
         wksRes.SetCell(ii, 3, vRes[ii + 3]);
      }
   }
   else
   {
      error_report("New worksheet can't be created.");
   }
```

- OnetTest_event1()

Update the function body, which will update results and show a conclusion at the bottom of the dialog according to the result. As settings change in the dialog, the **Null** edit box will be updated as the mean and hypothesis change, and the **Significance Level** edit box's value is checked.

```
   if( 0 == lstrcmp(lpcszNodeName, "mean") )
      _update_null(trGetN, true);
   if( 0 == lstrcmp(lpcszNodeName, "tail") )
      _update_null(trGetN);
   if( 0 == lstrcmp(lpcszNodeName, "siglevel") )
      bOKEnable = _check_sig_level(trGetN, strErrMsg);

   _update_GUI(trGetN);
   _update_strErr(trGetN, strErrMsg);

   return false;
```

- OnetTest_before_execute()

Update the function body, to show/hide or disable the controls in the dialog.

```
   StatTestWizCore* pstatwc = 
(StatTestWizCore*)get_xf_core_handler(trGetN.nXFCorePointer.nVal);
   ASSERT(pstatwc);
   trGetN.prob.Show = 1 - pstatwc->nStep;
   trGetN.prob.Enable = false;

   trGetN.null.Enable = false;
   trGetN.stat.Enable = false;
   trGetN.df.Enable = false;
   trGetN.tprob.Enable = false;
   trGetN.lcl.Enable = false;
   trGetN.ucl.Enable = false;
```

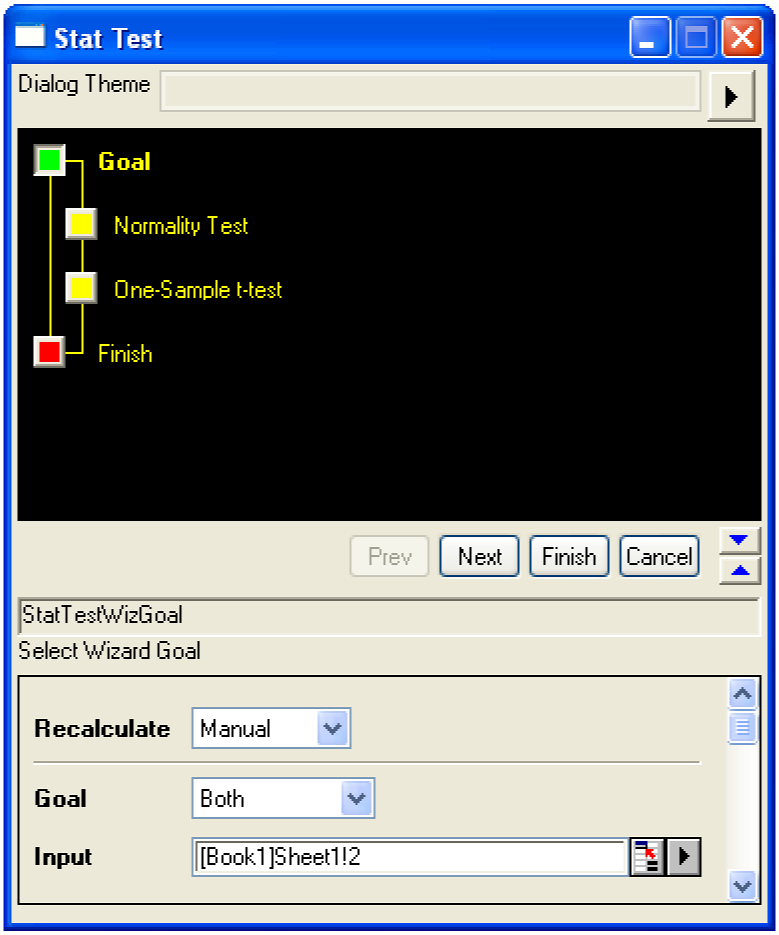
Click the **Compile** button to compile the file. Then click the **Return to Dialog** button to return to the X-Function Builder. Click the **Save OXF file** button to save the X-Function.

Close Origin. Then start Origin and you will notice that a new item **Stat Test** is added to Origin's menu **Statistics: Hypothesis Testing**.

11.9.7 How to Use the Wizard

The following example shows how to use the wizard.

1. Select a column in the worksheet.
2. Select **Statistics: Hypothesis Testing: Stat Test** from the Origin menu or type the command "StatTest -d" in the command window. The **Stat Test** wizard dialog will open.

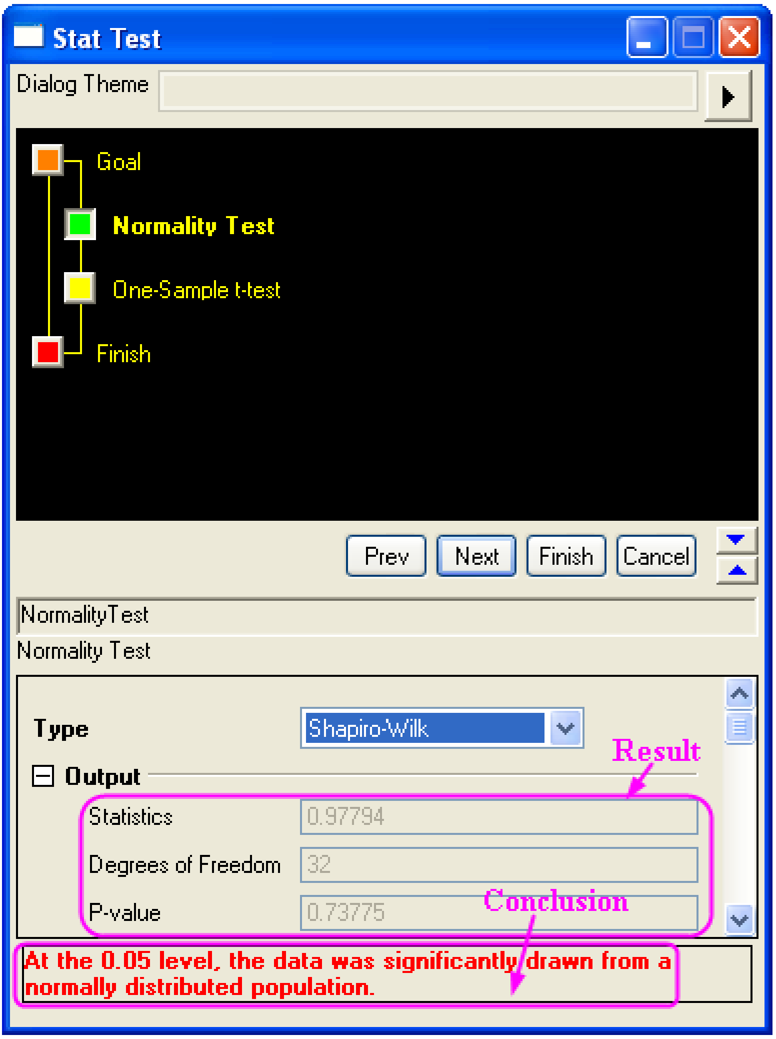


StatTestWizGoal

Select Wizard Goal

Recalculate	Manual
Goal	Both
Input	[Book1]Sheet1!2

3. Click the **Next** button. The **Normality Test** dialog is opened. The result is shown in the **Output** branch. A conclusion is drawn at the bottom of the dialog.



NormalityTest

Normality Test

Type: Shapiro-Wilk

Output

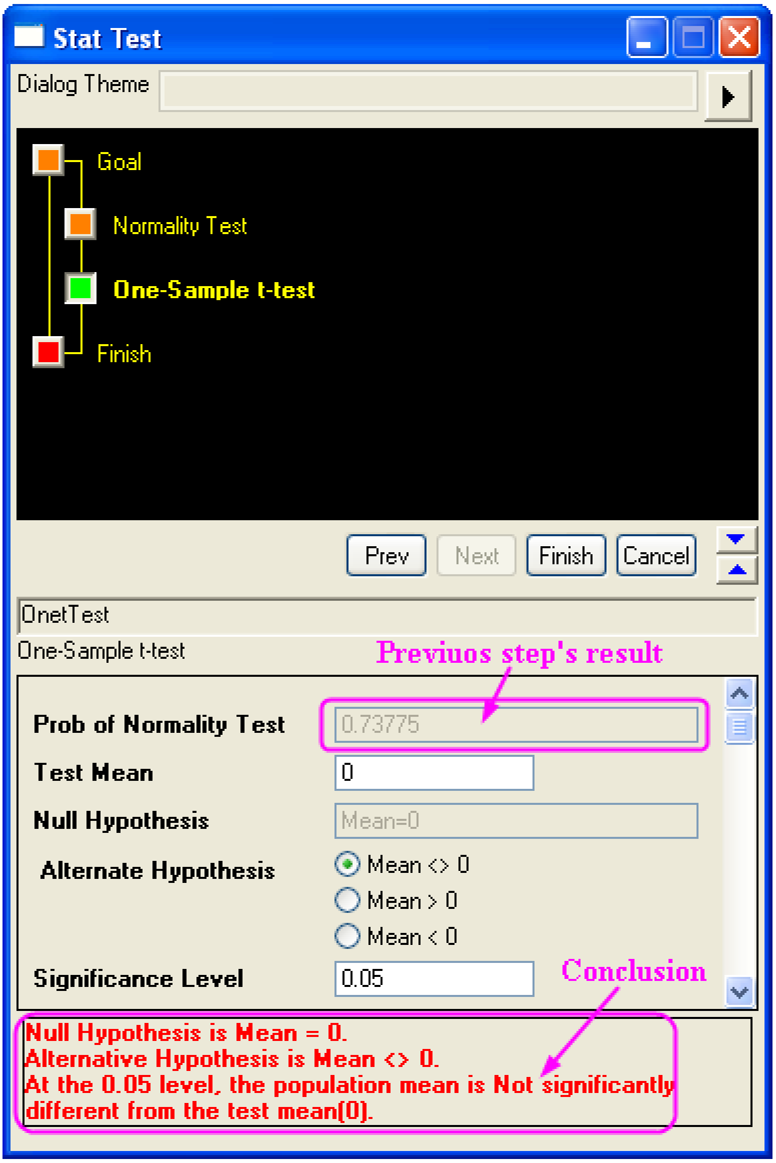
Statistics: 0.97794

Degrees of Freedom: 32

P-value: 0.73775

At the 0.05 level, the data was significantly drawn from a normally distributed population.

4. Click the **Next** button. The **One-Sample t-test** dialog is opened. The result is shown in the **Output** branch. A conclusion is drawn at the bottom of the dialog. Previous step's result of normality test is shown at the top. You can also change the setting in the dialog, and notice the



Prob of Normality Test: 0.73775

Test Mean: 0

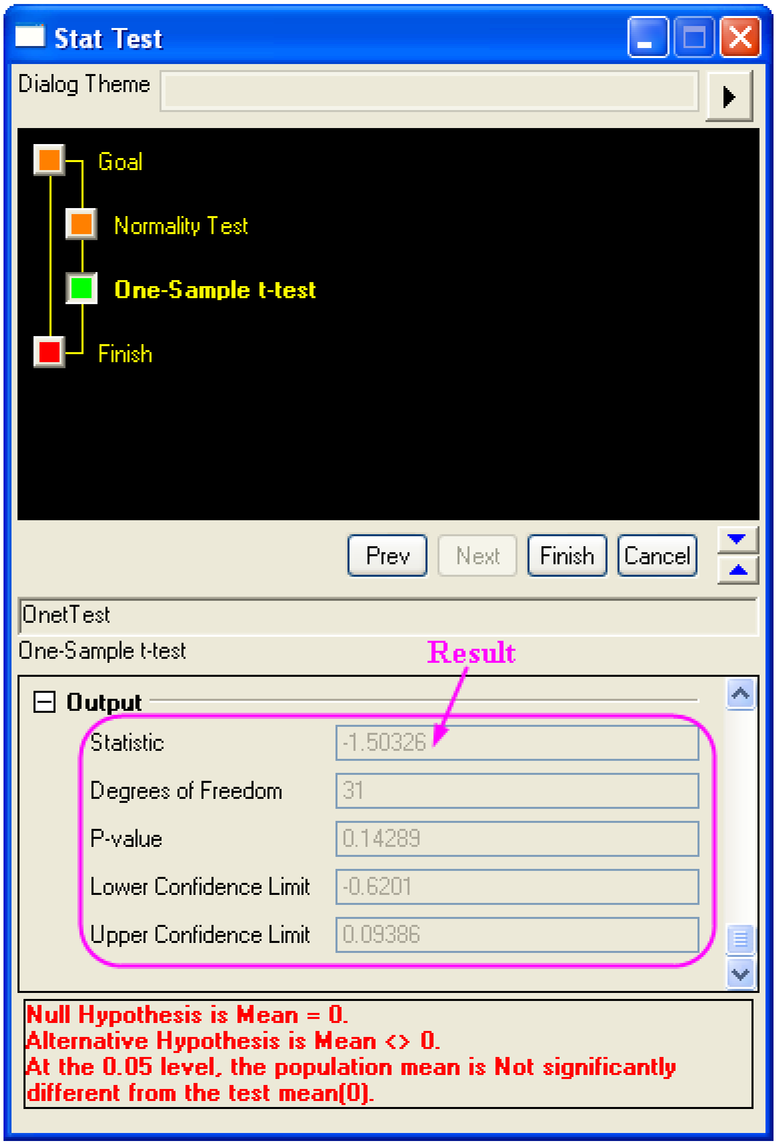
Null Hypothesis: Mean=0

Alternate Hypothesis: ● Mean <> 0 ○ Mean > 0 ○ Mean < 0

Significance Level: 0.05

Null Hypothesis is Mean = 0.
Alternative Hypothesis is Mean <> 0.
At the 0.05 level, the population mean is Not significantly different from the test mean(0).

result changes.



OnetTest

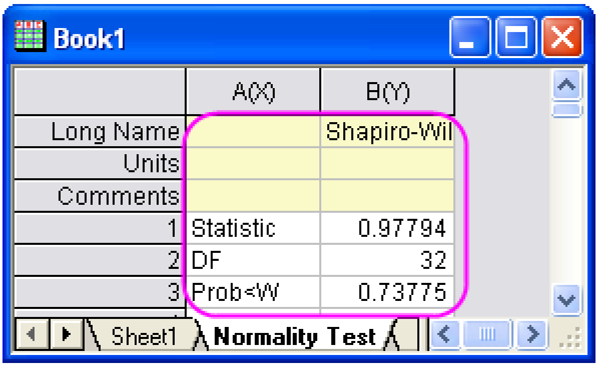
One-Sample t-test

Output

Statistic	-1.50326
Degrees of Freedom	31
P-value	0.14289
Lower Confidence Limit	-0.6201
Upper Confidence Limit	0.09386

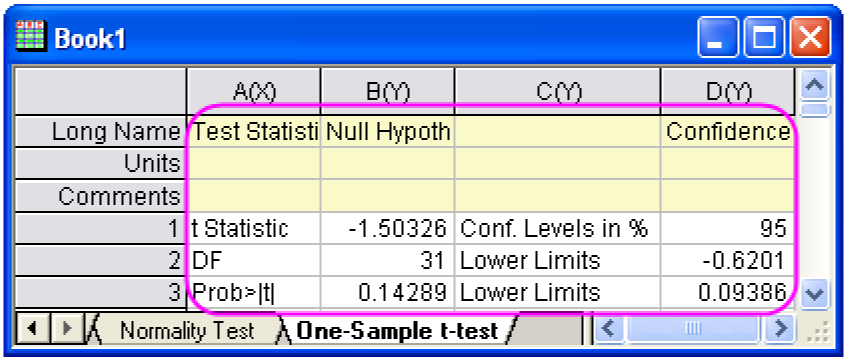
Null Hypothesis is Mean = 0.
Alternative Hypothesis is Mean <> 0.
At the 0.05 level, the population mean is Not significantly different from the test mean(0).

5. Click the **Finish** button to end the wizard. Two worksheets for results are created.



Book1

	A(X)	B(Y)
Long Name		Shapiro-Wil
Units		
Comments		
1	Statistic	0.97794
2	DF	32
3	Prob<W	0.73775



Book1

	A(X)	B(Y)	C(Y)	D(Y)
Long Name	Test Statisti	Null Hypoth		Confidence
Units				
Comments				
1	t Statistic	-1.50326	Conf. Levels in %	95
2	DF	31	Lower Limits	-0.6201
3	Prob>|t|	0.14289	Lower Limits	0.09386

11.10 How to share Origin C based LabTalk Functions

11.10.1 Introduction

This tutorial will show you how to share code with other users by distributing OPX files. In this example, only Origin C files are distributed. However, please note that OPX distribution can include any file types including (but not limited to) project files, templates and toolbars.

Minimum Origin Version Required: Origin 8 SR6

11.10.2 Tutorial

The following procedure will show you how to distribute your Origin C code to other Origin users. In this example, we will be packaging an Origin-C function in a file (*MyCode.c*) in a folder (*MyFunctions*) under the **User Files Folder** (**UFF**).

1. Create File Locations

   The **Source Path** for your files should be available on any computer your OPX is targeted for. The easiest way to do this is to make a subfolder of the Origin **UFF**. Any of the files and folders of the **UFF** can then be added to your OPX for distribution. So, create a subfolder named *MyFunctions* under the **UFF**.

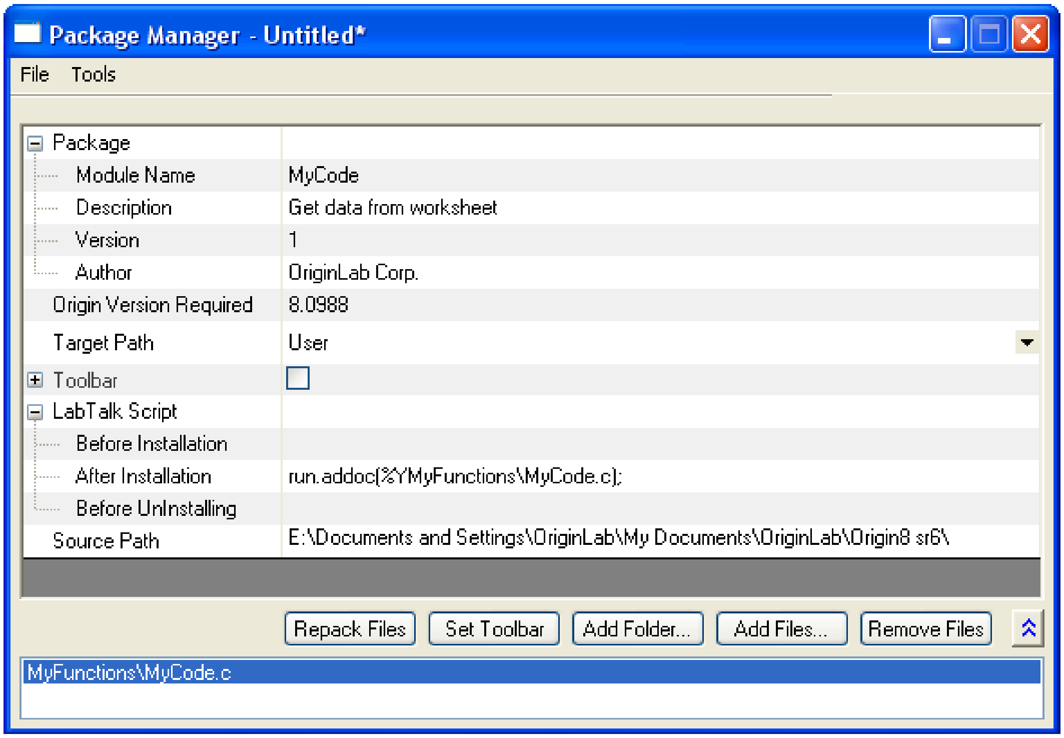
2. Copy Files to the Source Path

   Copy all the files to be packaged to the subfolder created in the last step. Here there is only a C file (*MyCode.c*). The function in this file is shown below.

```
void get_data_from_wks()
{
    Worksheet wks = Project.ActiveLayer();
    if( !wks )
    {
        out_str("Please keep a worksheet active with data");
        return;
    }

    // The following settings to set range as whole worksheet,
    // index offset is 0, -1 means last row/column.
    // Change r1, c1, r2, c2 to specify worksheet sub range,
    // for example, r1 = 0, c1 = 1, r2 = -1, c2 = 2 to
    // select all rows from column 2 to column 3.
    int r1 = 0; // first row
    int c1 = 0; // first column
    int r2 = -1; // last row
    int c2 = -1; // last column

    //construct a data range object from worksheet all data
    DataRange dr;
    dr.Add("X", wks, r1, c1, r2, c2);

    // get data from worksheet to vector by column one by one
    matrix mData;
    dr.GetData(mData); // get all data to matrix

    for(int nColIndex = 0; nColIndex < mData.GetNumCols(); nColIndex++)
    {
        vector vOneCol;
        mData.GetColumn(vOneCol, nColIndex);

        double min, max;
        vOneCol.GetMinMax(min, max);

        printf("Maximum value of %d column = %f\n", nColIndex+1, max);
    }
}
```

3. Create a Package

   Open the **Package Manager** by selecting **Tools: Package Manager...**. Then fill in the dialog as the following image shows.



⊟ Package	
Module Name	MyCode
Description	Get data from worksheet
Version	1
Author	OriginLab Corp.
Origin Version Required	8.0988
Target Path	User
⊞ Toolbar	□
⊟ LabTalk Script	
Before Installation	
After Installation	run.addoc(%YMyFunctions\MyCode.c);
Before UnInstalling	
Source Path	E:\Documents and Settings\OriginLab\My Documents\OriginLab\Origin8 sr6\

MyFunctions\MyCode.c

- The **Module Name** will appear as the name of the package for uninstall purposes.
- **Origin Version Required** means the minimum version of Origin required for this package. You want to enter the version number like 8.0988, and not like 8.0SR6.
- To force the Origin C source file to the System Folder Workspace, use the *run.addoc()* method in the **After Installation** branch of **LabTalk Script**. In this example, it looks like the following:

```
run.addoc(%YMyFunctions\MyCode.c);
```

- Click the **Add Files** or the **Add Folder** button and browse to and add any files required. In this example, browse to the subfolder (*MyFunctions*) under the **UFF** and add the C file (*MyCode.c*). Then the path of the **UFF** will show as the source path, and the files will list in the lower panel.

4. Save the Package

Select the **File: Save** menu of this **Package Manager** dialog. In the pop-up **Save As** dialog, enter a name (in this example, can be *MyCode*) for this package, and then save it as an OPX file.

5. Distribute the Package

Send the saved OPX file to other users. The user who gets this package can drag and drop the OPX onto Origin to install it, and then the functions in the C file are available.

- Drag and drop *MyCode.opx* onto Origin to install it. After installation, the function *get_data_from_wks* defined in *MyCode.c* can be used as a LabTalk command.

```
get_data_from_wks;
```