Additions and Improvements to LabTalk 6.1

The following LabTalk features have been implemented since the English Origin 6.0 release in June, 1999. Some features listed were available in Origin 6.0 patches.

Contents

Application Development

1) To simplify developing, editing, and debugging of LabTalk script files, Origin 6.1 includes a context-coloring LabTalk editor and debugger.
For information on using the LabTalk Editor and debugger (the debugger is available in OriginPro only), see the LabTalk 6.1 Help file.

2) When you press CTRL+SHIFT and select a menu command or click on a toolbar button, instead of outputting the script to the Script window if that command or toolbar button runs a script file section, Origin opens the script file in a new instance of the LabTalk Editor, with the cursor active in the appropriate script file section.

3) The **ed** object provides script access for the LabTalk Editor.
Methods:
ed.open( ) Opens a new LabTalk Editor window.
ed.open(*fileName*) Opens the specified script file in a new LabTalk Editor window.
ed.open(*fileName*,*sectionName*) Opens the specified script file at the specified section in a new LabTalk Editor window.
ed.openFile(*fileName*) Opens the specified script file in the active LabTalk Editor window.
ed.saveFile(*fileName*) Saves the current script in the active LabTalk Editor window to the *fileName* script file.

4) The **OPack** object is used to pack and unpack user-specified files.
For more information, see the LabTalk 6.1 Help file.

Control Flow

1) The following option for *object* has been added to the **doc -e *object* {*script*}** command:
LP Layer in graph windows only.

2) The following option has been added to the **doc** command:
-ef Same as the **-e** option, but only the current Project Explorer folder will be considered.
Note: This assumes that the Project Explorer view mode is set to **View Windows in Active Folder**. If the view mode is set to **View Windows in Active Folder & Subfolders**, then this command will consider the active folder and subfolders.

3) The following **system** object property has been added:
system.priority Controls the Origin EXEs Windows process priority: 64 = low, 32 = normal, 128 = high, 256 = realtime. Realtime is strongly discouraged.

4) The **switch** command now supports strings in the case statement. The string compare *is* case-sensitive.

Projects

1) When opening a project that was saved in Origin 4.1 and that contains a Text & Numeric column with mixed text and numeric values, a dialog box will open asking if the project was saved in the 32 or 16 bit version. The 32 bit version incorrectly saved mixed Text & Numeric data. When specifying 32 bit in the dialog box, Origin fixes the save problem by resaving the project in Origin 6.1 and then reopening the project. This control is provided with the **@MB** system variable.
0 = Opens the dialog box (default).
16 = Always assumes the project was saved in Origin 4.1 16 bit and thus opens the file with no changes.
32 = Always assumes the project was saved in Origin 4.1 32 bit and thus resaves the project to correct the problem.

2) The **@VDF** system variable is provided to report to the Script window the Origin version that the file was saved in.
0 = Do not report to the Script window.
1 = Report to the Script window. (If the project was saved in a version prior to 6.1, 0 is reported.)

Input/Output

1) The following **fdlog** properties have been added:
fdlog.multiOpen.comboList$ Contents of the combo box. Each item is separated with a |.
fdlog.multiOpen.comboName$ If empty, then hide the combo box. Otherwise, show the combo box and the label will equal the string.
fdlog.multiOpen.comboSel The default and the user selection from the combo box.

2) Origin provides a DLL named OMAIL.DLL that allows you to create a command object that provides basic email functionality. You can create a command object named **mail** using the following LabTalk script:
dll -a mail omail;
For more information, see the LabTalk 6.1 Help file.

3) Origin provides a DLL named OFTP.DLL that allows you to create a command object that provides basic FTP functions. You can create a command object named **oftp** using the following LabTalk script:
dll -a oftp oftp;
For more information, see the LabTalk 6.1 Help file.

4) Origin provides a DLL named OWKS2HTM.DLL that allows you to create a command object that provides worksheet to HTML export. You can create a command object named **Owks2Htm** using the following LabTalk script:
dll -a owks2htm owks2htm;
For more information, see the LabTalk 6.1 Help file.

5) A fixed width output is now supported allowing text and numbers to be output in a table-like format using the **type** command. Using the following notation, the output of the variable (represented by *Var*) will take up *i* number of characters no matter how long the variable is.
type $(Var,*,L$(*i*))

6) The **$( )** notation can be used together with C-like formatting conversion specifications - like the **%f** conversion specifier and the **.** decimal place modifier - to format output. The following options are available:
A. Output: SIGNED, Integer values (of decimal or integer value)
Input range: -2^31 to 2^31 -1
Example:
n = -247.56;
type "Value: $(n,%d)";
Value: -247
B. Option: f, e or E, g or G
Output: SIGNED, decimal, scientific, decimal or scientific
Input range: ±1e290 to ±1e-290
Examples:
n = 1.23456e5;
type "Values: $(n,%9.4f), $(n,%9.4E), $(n,%g)";
Values: 123456.0000, 1.2346E+005, 123456
---
n = 1.23456e6;
type "Values: $(n,%9.4f), $(n,%9.4E), $(n,%g)";
Values: 1234560.0000, 1.2346E+006, 1.23456e+006
C. Option: o, u, x, X
Output: UNSIGNED, Octal, Integer, hexadecimal, HEXADECIMAL
Input range: -2^31 to 2^32 -1
Example:
n = 65551;
type "Values: $(n,%o), $(n,%u), $(n,%X)";
Values: 200017, 65551, 1000F
(Negative values expressed as twos complements.)

Web Access

1) The following option has been added to the **win** command:
-aw *URL* Go to the *URL* web page. For example, **win -aw http://www.originlab.com**.

External Access

1) The following **DLL** command option has been added:
-a dllName Same as **-a *name dllName***, except that Origin reads the object name from the DLL's resource. The name must be in the string table with an ID value of 1.

Script Management

1) The **ini.check( )** method has been changed.
ini.check(*SectionName,KeyName*) Returns 0 if the *sectionName* or *keyName* exists, otherwise returns 1 and creates the *sectionName* and/or *keyName*. The *keyName* is set to 0 when created.

Additionally, the following properties and methods have been added to the **ini** object:
Properties:
ini.file$ A string with the file name to be accessed by the **ini** object. The Origin directory is assumed if no path is specified.
ini.key$ A string with the key name to be accessed by the **ini** object.
ini.section$ A string with the section name to be accessed by the **ini** object.
Methods:
ini.getStr(*stringVarLetter*, *defaultValue*) , **ini.setStr(*value*)** These methods can be used for accessing section and key names with spaces or other characters not allowed in LabTalk object/property names. These methods access the INI file, section, and key specified in the properties.

Timer

1) The **timer** command has an increased *cycleLength*. It can now run between 0.05 sec to 500 hours. Also, the **@TM** system variable indicates whether or not the timer is running.
1 = timer is running.
0 = timer is not running.

String Notation

1) To return the last token from the string **%B** into the string **%A**, use the following notation:
%A=%[%B,#+];
For example:
%A=Some File Format with ending extension (*.EPS);
%A=%[%A,#+];
%A=;
*.EPS

2) You can now add strings to worksheet cells set to Text and Numeric by enclosing the string in quotation marks. For example **%(Data1,1,1)="*string*"** will place *string* in the first row of the first column of Data1.

Worksheets, Datasets, Matrices

1) The following **create** command options have been modified or added:
-wd *numPoints* [*colName*(*s*)] Create the datasets but not the worksheet. Set datasets to Text & Numeric.
-wdn *numPoints* [*colName*(*s*)] Create the datasets but not the worksheet. Set datasets to Numeric (not Text & Numeric).
-wh *numPoints* [*colName*(*s*)] Create a hidden worksheet with the datasets set to Text & Numeric.
-whn *numPoints* [*colName*(*s*)] Create a hidden worksheet with the datasets set to Numeric.
-wn *numPoints* [*colName*(*s*)] Create a worksheet with the datasets set to Numeric.

2) The following **delete** command option has been added:
-as Delete all datasets that are not in any worksheet and not used in any graphs.

3) If you execute the **limit** command while a graph with Speed Mode ON is active (on the Size/Speed tab of the layer's Plot Details dialog box), then the results will only include the actual displayed values.

4) To execute the **limit** command on a Y dataset that does not have an associated X dataset, use **limit -s *dataset***. This command option will set the Y limits only.

5) The following options have been added to the **win** command:
-cd *winName* Close the worksheet and delete all the data.
-tx wks excel Open a new Excel workbook.

6) The **cell(*colNum*,*rowNum*)** function is provided to get or set values in the active worksheet or matrix. The *col* value is the numeric value of the column and *row* is the row number. See the following item for related information. (Note: In Origin 6.1 **SR1**, the notation is **cell(*rowNum*,*colNum*)**.

7) When assigning text values to a Text & Numeric column using LabTalk, instead of using the **%(%H,*columnNum*,*rowNum*)** substitution notation, use the **col(*columnNum*)[*rowNum*]$** function notation or use the **cell(*columnNum*,*rowNum*)$** function notation. (Note: In Origin 6.1 **SR1**, the notation is **cell(*rowNum*,*colNum*)**.

8) The **NewFunction** macro has been redefined to prevent duplicate function names and possible loss of functions. The redefinition includes a new command:
list -sn *NameBase* [*strChar*]
This command will start searching for *NameBase* and will begin enumerating from 1. It will search for the next available name that is not in conflict with any dataset. It will put the result in the given *strChar*. If *strChar* is not included, it will be saved into **%A**.
For the **NewFunction** macro, the **list -sn F B;** command is used. Origin searches for all datasets F*n*, starting with F1, F2, ... etc. The first unused F*n* found is put in the **%B** string variable.
As another example, if you have a new default project with a data1 worksheet, **list -sn data B;** will put Data2 into **%B**.

9) When editing worksheet (or matrix) cell values, the left and right arrow keys will no longer exit the edit mode when you go past the first or last cell character. This option is controlled by the **@WE** system variable.
0 = do not exit edit mode (default).
1 = exit the edit mode and proceed to the left or right cell.

10) The **@ZL** system variable is provided to control how Origin treats data with leading zeros in worksheet cells.
1 = check for leading zeros in numeric testing. Thus, 02 will be considered text.
0 = treat values with leading zeros as numeric. Thus, 02 will be considered as 2.

11) Three new system variables are available to control how masked data is treated with vector arithmetic, cell access, and data selection:
@MV Controls mask options on vector arithmetic (for example, **col(2) = col(3) + col(4)**). 0 = ignore masked data (default). 1 = use masked data on the right side of the equation.
@MD Controls mask options on cell access using ***worksheetName_columnName*[*i*]** and **cell(*j*,*i*)**. 0 = ignore masked data. 1 = use masked data (default).
@MS Controls mask options related to data selection for operation on. 0 = when masked data

is part of a selection, it will be interpreted as selected (default). 1 = when masked data is part of a selection, it will be interpreted as not selected.

12) You can now add strings to worksheet cells set to Text and Numeric by enclosing the string in quotation marks. For example **%(Data1,1,1)="*string*"** will place *string* in the first row of the first column of Data1.

13) The **mat** object has been updated for computing the X and Y projections and for matrix profiling.
Properties:
mat.edgeView.xeCol$ Used with the **mat.edgeview( )** method. This is the X column in the **mat.edgeview.xeWksName$** worksheet.
mat.edgeView.xeWksName$ Used with the **mat.edgeview( )** method. This is the worksheet where the X projection is stored.
mat.edgeView.yeCol$ Used with the **mat.edgeview( )** method. This is the worksheet where the Y projection is stored.
mat.edgeView.yeWksName$ Used with the **mat.edgeview( )** method. This is the worksheet where the Y projection is stored.
mat.edgeView.zxeCol$ Used with the **mat.edgeview( )** method. This is the Y column in the **mat.edgeview.xeWksName$** worksheet. This column contains the maximum Y along that X.
mat.edgeView.zxsCol$ Used with the **mat.edgeview( )** method. This is a column in the **mat.edgeview.xeWksName$** worksheet. This column contains the sum of all Ys along that X.
mat.edgeView.zyeCol$ Used with the **mat.edgeview( )** method. This is the X column in the **mat.edgeview.xeWksName$** worksheet. This column contains the maximum X along that Y.
mat.edgeView.zysCol$ Used with the **mat.edgeview( )** method. This is a column in the **mat.edgeview.xeWksName$** worksheet. This column contains the sum of all Xs along that Y.
mat.profile.angle Used with the **mat.profile( )** method - for a profile along a line with a preset slope. This is the angle with X axis (in radians). The default value is PI/4.
mat.profile.hSection Used with the **mat.profile( )** method. This is the number of points in the horizontal profile.
mat.profile.hwksName Used with the **mat.profile( )** method. This is the worksheet that stores the horizontal profile.
mat.profile.hXCol Used with the **mat.profile( )** method. This is the worksheet column that stores the X horizontal profile.
mat.profile.hZCol Used with the **mat.profile( )** method. This is the worksheet column that stores the Z horizontal profile.
mat.profile.sSection Used with the **mat.profile( )** method. This is the number of points in the slanted profile.
mat.profile.vSection Used with the **mat.profile( )** method. This is the number of points in the vertical profile.
mat.profile.vWksName Used with the **mat.profile( )** method. This is the worksheet that stores the vertical profile.
mat.profile.vYCol Used with the **mat.profile( )** method. This is the worksheet column that stores the Y vertical profile.
mat.profile.vZCol Used with the **mat.profile( )** method. This is the worksheet column that stores the Z vertical profile.
mat.wksName$ Name of the active worksheet. (Used with the **mat.profile( )** method. This

is the worksheet that stores the slanted profile.)
mat.xCol The X column number in the worksheet. (Used with the **mat.profile( )** method. This is the worksheet column that stores the X slanted profile.)
mat.yCol The Y column number in the worksheet. (Used with the **mat.profile( )** method. This is the worksheet column that stores the Y slanted profile.)
mat.zCol The Z column number in the worksheet. (Used with the **mat.profile( )** method. This is the worksheet column that stores the Z slanted profile.)
Methods:
mat.edgeView( ) Plot the maximum values of X along a given Y, and Y along a given X. You can also specify a column (optional) to compute and store the sum of all Y along a particular X and the sum of all X along a particular Y.
mat.profile(*x1*,*y1*) Provides three profiles: 1) The horizontal profile y = y1. 2) The vertical profile x = x1. 3) The profile along a line with a preset slope passing through (x1,y1). Set the angle with the X axis in radians using the **mat.profile.angle** property. The default angle is PI/4. The profiles are stored in the following worksheets: 1) Horizontal: **mat.profile.hwksname**, **mat.profile.hXcol**, **mat.profile.hZcol**. 2) Vertical: **mat.profile.vwksname**, **mat.profile.vYcol**, **mat.profile.vZcol**. 3) Slanted: **mat.wksname**, **mat.xcol**, **mat.ycol**, **mat.zcol**. The number of points in each profile can be accessed from **mat.profile.hsection**, **mat.profile.vsection**, and **mat.profile.ssection**.
mat.segment(*x1*,*y1*,*x2*,*y2*) Find the profile between (*x1*,*y1*) and (*x2*,*y2*). The results are written to **mat.wksName$**.

Importing and Exporting Data

1) Origin provides a DLL named ODBC60.DLL that allows you to create an ODBC command object whose properties and methods connect Origin to Microsoft's ODBC interface. You can create a command object named **db** using the following LabTalk script:
dll -a db odbc60;
See the LabTalk 6.1 Help file for more information.

2) The **image** object provides script access for importing a raster graphic image file into a matrix and for exporting the matrix to a raster graphic image file.
For more information, see the LabTalk 6.1 Help file.

3) Origin now provides a **worksheet -kl *numLines*** command option to control how many lines of the skipped header lines will be put into the column labels during ASCII import. The default value is 7.

Graphs

1) When using the **draw [*option*(s)] -l [-h / -v] *value*** command to create or alter a line, consider the following additions:
To create a *movable* line, use **-lm** instead of **-l**.
By default, lines and arrows are attached to the Layer and Scales. To disable this, set **@AR** = 0.

2) The **layer -e** command option has been updated:
-e *dataset* Remove *dataset* from the active layer - but do not remove the data plot style holder.

3) A new **@ILC** system variable initializes the 'clip data to frame' settings.
1 = clip data to frame (axes).
2 = clip data to axes break region.
3 = clip data to both.

4) A new **@ILD** system variable initializes the 'show data on top of axes and grid' settings.
1 = draw grids after data.
2 = draw axes after data.
Note that these are bits so they can be combined.

5) Origin now provides improved line and arrow scaling. Lines and arrows are now attached to the Layer and Scales by default (Label Control dialog box). Additionally, lines and arrows will now always use real XY values for each end point. End points can be accessed by script:
***objectName*.x1** = start X position
***objectName*.x2** = end X position
***objectName*.y1** = start Y position
***objectName*.y2** = end Y position
Note also that the ***objectName*.x** and ***objectName*.y** properties have been channeled to ***objectName*.x1** and ***objectName*.y1**, so you can continue to use ***objectName*.x** to move a vertical line or ***objectName*.y** to move a horizontal line.

6) The **set** command now includes options for controlling the box gap and border width in bar and column graphs.
-vg *percentGap* Set the box gap.
-vw *borderWidth* Set the border width in points.

7) The color scale object, named Spectrum1, now has a **Spectrum1.SetDeafult( )** method. Origin only uses this method when loading templates or when adding a color scale to a graph that does not have one (for example, for version 5.0-saved graphs and templates).

8) To control the gap between symbols when using "overlapped points offset plotting", use the following system variables:
@OS 1 = display overlapped points both horizontally and vertically about the XY value. 0 = display overlapped points only horizontally about the XY value (default).
@OSG This variable controls the percent of the symbol size to use as the gap between the center of adjacent symbols. For example, when **@OSG** = 125 (default), then the gap between adjacent symbols (as measured from the center of the symbols) is 125% of the symbol size.

9) To specify the width of an axis break, use the **@AB** system variable. This variable sets the axis break as 1/100% of the total axis length. For example, if the current value of the **@AB** system variable is 172, then the width of the axis break is (172/100)% or 1.72% of the total axis length.

10) The **@LP** system variable is provided to control the maximum number of points used in the GDI call to draw line plots, 3D XYY plots, and some other plot types. Some video drivers can not handle large numbers passed as arguments in the GDI call, so Origin breaks the drawing up into several sections that are each **@LP** points large.
The **@LP** value is saved between Origin sessions in the MaxPolyLinePoints keyword in the [Display] section of the ORIGIN.INI file.

11) The **@SF** system variable is provided as a scaling factor to control the legend symbol size.

Copying and Exporting Graphs

1) The **image** object provides script access for exporting graphs to graphic files. It also provides script access for importing a raster graphic image file into a matrix and for exporting the matrix to a raster graphic image file.
For more information, see the LabTalk 6.1 Help file.

2) The following **system.fileExt** method has been added:
system.fileExt.getType(*extension*) Returns the index of the **system.fileExt.type*n*$** property that represents the file type specified in *extension*. If no file type represents the extension then zero is returned. Example *extension* is BMP, EPS, JPG, etc. This is not case sensitive.

Analysis and Fitting

1) If you execute the **limit** command while a graph with Speed Mode ON is active (on the Size/Speed tab of the layer's Plot Details dialog box), then the results will only include the actual displayed values.

2) To execute the **limit** command on a Y dataset that does not have an associated X dataset, use **limit -s *dataset***. This command option will set the Y limits only.

3) The **fft.exponentSign** property has been changed to **fft.convention**.
fft.convention Switch the sign used in calculating FFT.
0= negative exponent for forward FFT, and positive for backward FFT. This is the Electronic Engineering convention.
1= positive exponent for forward FFT, and negative for backward FFT. This is the Physics convention.

4) When the **curve.SGSmooth( )** or **curve.SGDeriv( )** method is run, Origin will check if **curve.smoothRightPts** + **curve.smoothLeftPts** + 1 is equal to **curve.smoothPts**. If not, Origin will reassign the left and right windows as half of **curve.smoothPts** (+1).

5) When performing vector operations on datasets using the *dataset operator dataset* statement (for example, **data3_b - data2_b**), interpolation is used to find the Y values of the second dataset (**data2_b**) using the X values of the first dataset (**data3_b**). In Origin 6.1, interpolation is performed within the domain of the second dataset only. In previous versions, interpolation was performed within the domain of the second dataset plus one point on either side. To return to the pre-6.1 behavior, set **@IE** = 1. The new default is **@IE** = 0.

6) An **fft.corr.posLag** property is now available which corrects previous problems with autocorrelation.
1 = display positive lag values only, up to the dataset size.
0 = display all lag values up to the padded dataset size multiplied by 2.

7) The **dotool -dh *toolname*** command allows you to open a tool without displaying the Data Display tool.

8) The **min( )** and **max( )** functions have been added to return the minimum and maximum values from a group of arguments. For example, **y=min(1,2,3,4)**. Up to 10 arguments are supported.

Customization

1) The **type** command provides script control over the display of reminder messages. When a reminder message displays and the user selects "do not show again", Origin writes a keyword entry to the [ReminderMessages] section of the ORIGIN.INI file. If the user clicked OK or Yes in the reminder message, then the keyword is set to 1. If the user clicked Cancel or No, then the keyword is set to 0. The keywords are:
ProjectAppendCreateNewFolder
SpeedModeCopyPageExport
SpeedModeScreenDisplay
EPSUsePrint
CustomPageSizeOrientationNoChange
Win2000RegistryWriteFailed
TernryPlotNormalization
The **type** command options are:
type -m *keyword* Display the *keyword* reminder message.
type -mr [*keyword*] Delete the *keyword* reminder message entry from the ORIGIN.INI file. If *keyword* is not included, this command option deletes all the reminder message keyword entries in the ORIGIN.INI file.
type -mb value Begin the reminder message override mode and set the reminder message return values to *value*, which as 1 for OK and Yes and 0 for Cancel and No.
type -me End the reminder message override mode. You must use this command, otherwise no reminder messages will display.

2) The **@SD** system variable is provided to control the number of significant digits displayed when mathematical operations are performed in the Script window. This variable is defaulted to 7 and can be set to any integer between 1 and 15.

3) The **system.toolbar** object is provided to control the display of toolbars.
Properties:
system.toolbar.names$ A list of the names of all the toolbars.
Methods:
system.toolbar.create(*category*[,*toolbarName*]) Open a toolbar from a user-defined category.
system.toolbar.delete(*toolbarName*) Delete a toolbar, including from the Registry.
system.toolbar.exists(*toolbarName*) Returns 1 if the toolbar exists. Otherwise, returns 0.
system.toolbar.isOpen(*toolbarName*) Returns 1 if the toolbar is open. Otherwise, returns 0.
system.toolbar.open(*toolbarName*) Open a toolbar. Returns 0 if OK. Otherwise, returns 1.
system.toolbar.close(*toolbarName*) Close a toolbar. Returns 0 if OK. Otherwise, returns 1.

Miscellaneous

1) The **exist(*name*)** function now works for tools (*name* = 7), allowing you to check whether a tool is open or closed.

2) The **exist(*name*)** function will now return the size of the given file. If the file does not exist, the function returns -1.

3) The **system.path.findtemp(*str*,*NewSub*<,*.*>)** method will check if a *NewSub* folder is present in the Windows Temp path. The Windows Temp path is specified by the string variable *str*. If not present, it will create the folder. If the *NewSub* folder is already present and the optional third argument is provided, then files of type *.* (any wild card, such as *.OPJ) will be deleted from the folder. Note: *.* will delete all files.