ModifyBrowser

ModifyBrowser [/M] [***keyword = value*** [, ***keyword = value*** …]]

The ModifyBrowser operation modifies the state of the Data Browser according to the specified keywords.

Documentation for the ModifyBrowser operation is available in the Igor online help files only. In Igor, execute:

```
DisplayHelpTopic "ModifyBrowser"
```

ModifyCamera

ModifyCamera [***flags***] **[*keywords***]

The ModifyCamera operation modifies the properties of a camera window.

Documentation for the ModifyCamera operation is available in the Igor online help files only. In Igor, execute:

```
DisplayHelpTopic "ModifyCamera"
```

ModifyContour

ModifyContour [**/W=*winName***]***contourInstanceName*, *keyword=value***
[, ***keyword=value***…]

The ModifyContour operation modifies the number, Z value and appearance of the contour level traces associated with *contourInstanceName*.

contourInstanceName is a name derived from the name of the wave that provides the Z data values. It is usually just the name of the wave, but may have #1, #2, etc. added to it in the unlikely event that the same Z wave is contoured more than once in the same graph.

contourInstanceName can also take the form of a null name and instance number to affect the instanceth contour plot. That is,

```
ModifyContour ''#1
```

modifies the appearance of the second contour plot in the top graph, no matter what the contour plot names are. Note: Two single quotes, not a double quote.

The number of contour level traces and their Z values are set by the *autoLevels*, *manLevels*, and *moreLevels* keywords, described in the **Parameters** section. Normally, you will use either *autoLevels* or *manLevels*, and then optionally generate additional levels using *moreLevels*.

Parameters

Each parameter has the syntax

```
keyword = value
```

and is applied to all of the contour level traces associated with *contourInstanceName*.

To modify an individual contour level trace, use **ModifyGraph**.

allocationFactor=*factor* In rare situations that usually involve spatial degeneracy, the Voronoi interpolation algorithm may need additional memory. You can use the allocationFactor keyword with an integer factor greater than 1 to increase the memory allocation.

allocationFactor was added in Igor Pro 9.00.

autoLevels= {*minLevel*, *maxLevel*, *numLevels*}

Controls automatic determination of contour levels.

If *numLevels* is zero, no automatic levels are generated. If it is nonzero, it specifies the desired number of automatic contour levels.

minLevel specifies the minimum contour level and *maxLevel* specifies the maximum contour level. The values that you specify are an approximate guide for Igor to use in determining the actual levels.

However, if *minLevel* or *maxLevel* is * (asterisk symbol), Igor uses the minimum or maximum value of the Z data for the corresponding contour level.

Using the autoLevels keyword cancels the effect of any previous autoLevels or manLevels keyword.

When you first append a contour plot to a graph, default contour levels are generated by the default setting `autoLevels={*,*,11}`.

boundary=*b*

Draws an outline around the XY domain of the contour data. For a matrix, this draws a rectangle showing the minimum and maximum X and Y values. For XYZ triples, the outline is a polygon enclosing the outside edges of the Delaunay Triangulation. Like the contour lines, the boundary is drawn using a graph trace, whose name is usually something like "`contourInstanceName = boundary`".

b=0: Hides the data boundary (default).

b=1: Shows the data boundary.

cIndexFill= *matrixWave*

Sets contour fills to use a color index wave when automatic fill is on (see the fill keyword).

cIndexFill works the same as the cIndexLines keyword which controls the colors of the contour level traces.

See **Contour Fills** on page II-373 for more information.

cIndexFill was added in Igor Pro 7.00.

cIndexLines= *matrixWave*

Sets the Z value mapping mode such that contour line colors are determined by doing a lookup in the specified matrix wave.

matrixWave is a 3 column wave that contains red, green, and blue values from 0 to 65535. (The matrix can actually have more than three columns. Any extra columns are ignored.)

The color for a the contour line at Z=*z* is determined by finding the RGB values in the row of *matrixWave* whose scaled X index is *z*. In other words, the red value is *matrixWave*(*z*)[0], the green value is *matrixWave*(*z*)[1] and the blue value is *matrixWave*(*z*)[2].

If *matrixWave* has default X scaling, where the scaled X index equals the point number, then row 0 contains the color for Z=0, row 1 contains the color for Z=1, etc.

If you use cIndexLines, you must not use ctabLines or rgbLines in the same command.

cTabFill= {*zMin*, *zMax*, *ctName*, *mode*}

Sets contour fills to use a color table when automatic fill is on (see the fill keyword).

cTabFill works the same as the ctabLines keyword which controls the colors of the contour level traces.

See **Contour Fills** on page II-373 for more information.

cTabFill was added in Igor Pro 7.00.

ctabLines={*zMin*, *zMax*, *ctName*, *mode*}

Sets the Z value mapping mode such that contour line colors are determined by doing a lookup in the specified color table. *zMin* is mapped to the first color in the color table. *zMax* is mapped to the last color. Z values between the min and max are linearly mapped to the colors between the first and last in the color table.

You can enter * (an asterisk) for *zMin* and *zMax*, which uses the minimum and maximum Z values of the data. The default is {*,*,Rainbow}.

Set parameter *mode* to 1 to reverse the color table; zero or missing does not reverse the color table.

ctName can be any color table name returned by the **CTabList** function, such as Grays or Rainbow (see **Image Color Tables** on page II-392) or the name of a 3 column or 4 column color table wave (see **Color Table Waves** on page II-399).

A color table wave name supplied to `ctabLines` must not be the name of a built-in color table (see **CTabList**). A 3 column or 4 column color table wave must have values that range between 0 and 65535. Column 0 is red, 1 is green, and 2 is blue. In column 3 a value of 65535 is opaque, and 0 is fully transparent.

If you use ctabLines, you must not use cIndexLines or rgbLines in the same command.

equalVoronoiDistances=*e*

Normally the x range and y range of the data are each normalized to a 0-1 range separately to generate the Voronoi triangulation. Voronoi triangulation is a distance-based ("nearest neighbor") algorithm that may benefit from scaling the X and Y ranges together to avoid numerical problems that occur when the triangles become very thin because of widely differing x and y ranges.

e=0: The x and y ranges are scaled individually to the 0-1 range (default).

e=1: The x and y ranges are scaled so that that maximum range of x or y is scaled to the 0-1 range, and the other is proportionally smaller. For example, if yMax-yMin = 1000 and xMax-xMin = 5, then the y range is scaled to 0-1 and the y range is scaled to 5/1000 = 0 - 0.005.

The equalVoronoiDistances keyword is allowed only for XYZ contour plots.

fill=*f*

Controls the automatic filling of contour levels.

f=0: Turns automatic fill off. Default.

f=1: Turns automatic fill on.

See **Contour Fills** on page II-373 for more information.

fill was added in Igor Pro 7.00.

interpolate=*i*

XYZ contours can be interpolated to increase the apparent resolution, resulting in smoother contour lines.

This keyword is allowed only for XYZ contours, created by **AppendXYZContour**.

i=0: Linear interpolation (default). This means that only the original Delaunay triangulation generates contour lines.

i=1: Four times the resolution generates a smoother set of contour lines. As expected, this takes longer than Linear interpolation.

i=2: Sixteen times the resolution generates a much smoother set of contour lines. This is rather slow.

The interpolate parameter can be up to 8. Each time you increase *i* by one, you quadruple the apparent resolution and get smoother contour lines at the expense of computation time. Values of *i* greater than two are impractical because of the computation time required.

labelBkg=(*r*,*g*,*b*[,*a*])

Sets the background color for all contour level labels to the specified color. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**.

labelBkg=*b*

Controls the background color of contour labels.

b=0: Uses each label's individual background color, as set via the Modify Annotation dialog.

b=1: Makes all contour level labels transparent.

b=2: Uses the plot area background color as the label background color (default).

b=3 Uses the window background color as the label background color.

labelDigits=*d*

d is the number of digits after the decimal point when using labelFormat=3 or labelFormat=5.

labelFont=*fontName*

Default; specifies the font to use for contour level labels. If you pass "" for *fontName*, it will use the graph font (set via the Modify Graph dialog) for contour labels.

labelFormat=*l*

Controls the formatting of contour labels. See the **printf** operation for a discussion of formatting.

l=0: Uses general format that is suitable for most data. This is equivalent to "%<sigDigits>g".

l=1: Uses integer format, equivalent to "%<sigDigits>d". This rounds fractional values.

l=3: Uses fixed point format, equivalent to "%<decimalDigits>f".

l=5 Uses exponential format, equivalent to "%<decimalDigits>e".

labelFSize=*s*

Specifies the font size of contour labels in points. For example, use labelSize=12 for 12 point type. The default value is 0, which chooses the size automatically based on the size of the graph.

labelFStyle=*n*

n is a bitwise parameter with each bit controlling one aspect of the font style for the contour level labels. The default is 0, plain text.

Bit 0: Bold

Bit 1: Italic

Bit 2: Underline

Bit 4: Strikethrough

See **Setting Bit Parameters** on page IV-12 for details about bit settings.

labelHV=*hv*

Specifies the contour label orientation.

If *hv* is 3, 4, 5, or 6, the contour label's text rotates whenever it is redrawn, usually when the underlying contour data changes, the graph is resized, or the label is reattached to a new contour trace point.

hv=0: Horizontal contour level labels.

hv=1: Vertical contour level labels.

hv=2: Horizontal or vertical contour level labels, depending on the slope of the contour line.

hv=3: Tangent to the contour line.

hv=4: Tangent to the contour line, snaps to vertical or horizontal if within 2 degrees of vertical or horizontal (default).

hv=5: Perpendicular to the contour line.

hv=6: Perpendicular to the contour line, snaps to vertical or horizontal if within 2 degrees of vertical or horizontal.

labelRGB=(*r*,*g*,*b*[,*a*])

Sets the text color for all contour level labels. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**. The default is opaque black.

labels=*l*

Controls the display of contour labels.

l=0: Hides contour level labels.

l=1: Leaves any contour level labels in place but stops updating them and stops generation of new labels.

l=2: Generates or updates labels for the existing contour levels and window size when the command executes, but disables further updating of labels when window size or contour plot changes. This is the recommended setting if updating the labels takes long enough to annoy you.

l=3: Default; generates labels for all contour levels whenever the contoured data changes but not when the window size changes. If you resize the graph, the labels may overlap or be too sparse.

l=4: Generates labels for all contour levels whenever the contoured data, contour levels, axis range, or the graph size changes. (Actually, there are too many causes to list here. If all this update annoys you, use labels=2 “update once, now”.)

labelSigDigits=*d*

d is the number of significant digits when labelFormat=0 is used.

logLines= 1 or 0

0 sets the default linearly-spaced contour line colors.

1 turns on logarithmically-spaced line colors. This requires that the contour levels values be greater than 0 to display correctly.

Affects line color only when the cIndexLines or ctabLines parameter is used.

logLines does not affect the contour levels. To assign logarithmically-spaced contour levels, use the moreLevels parameter and disable autoLevels, for example:

```
ModifyContour ''#0, autoLevels={*,*,0} // No auto levels
ModifyContour ''#0, moreLevels=0
ModifyContour ''#0, moreLevels={1e-07,1e-06,1e-05,1e-04}
```

manLevels= {*firstLevel*, *increment*, *numLevels*}

Explicitly specifies contour levels. ModifyContour will generate *numLevels* contour levels, evenly spaced starting from *firstLevel* and stepping by *increment*.

manLevels cancels the effect of any previous manLevels or autoLevels settings.

manLevels= *manLevelsWave*

Explicitly specifies contour levels. ModifyContour will generate contour levels at the values in *manLevelsWave*.

manLevels cancels the effect of any previous manLevels or autoLevels settings.

moreLevels= {*level*, *level* …}

Explicitly specifies contour levels. ModifyContour will generate a contour trace for each of the listed levels. The maximum number of levels that you can specify in a single command is the 50. However, you can concatenate any number of ModifyContour moreLevels commands. moreLevels adds levels in addition to any specified by manLevels or autoLevels. It does not override other parameters.

moreLevels=0: Removes all levels generated by previous moreLevels settings.

nullValue=z*Value* This keyword only affects the behavior of the **ContourZ** function. It is allowed only for XYZ contours, created by **AppendXYZContour**.

By default, ContourZ treats data outside the domain of the contour as NaN and so returns NaN if you ask for a contour value outside that domain.

The nullValue keyword allows you to change the default behavior to make ContourZ treat values outside the domain as the specified zValue.

nullValueAuto This keyword only affects the behavior of the **ContourZ** function. It is allowed only for XYZ contours, created by **AppendXYZContour**.

nullValueAuto acts like nullValue=z*Value* with z*Value* automatically set to the minimum value in the Z wave minus 1.

See the nullValue keyword for details.

To turn nullValueAuto off and return the contour to the default state, execute:

```
ModifyContour <contourInstanceName>, nullValue=NaN
```

perturbation=*p* Enable or disable perturbation (alteration) of the x and y values by a miniscule amount to improve the natural neighbor triangulation of XYZ contours.

- *p*=0: Disables perturbation, preserving the original x and y values unchanged.
- *p*=1: Enables x/y perturbation (default). The values are shifted by random values less than +/-0.000005 times the x and y domain extents.

You can observe the perturbed x/y coordinates in the triangulation trace added by ModifyContour triangulation=1.

The perturbation keyword is allowed only for XYZ contour plots.

rgbFill=(*r*,*g*,*b*[,*a*]) Specifies red, green, and blue values for all contour fills. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**.

If you use rgbFill, you must not use cIndexFill or ctabFill in the same command.

rgbLines=(*r*,*g*,*b*[,*a*]) Specifies red, green, and blue values for all contour lines. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**.

If you use rgbLines, you must not use cIndexLines or ctabLines in the same command.

triangulation=*t* Draws the Delaunay Triangulation. As part of the XYZ contouring algorithm, the XY domain is subdivided into triangles in a process called Delaunay Triangulation. Like the contour lines, the triangulation is drawn using a graph trace, whose name is usually something like "contourInstanceName =triangulation".

The triangulation keyword is allowed only for XYZ contours, created by **AppendXYZContour**.

- *t*=0: Hides the Delaunay triangulation (default).
- *t*=1: Shows the Delaunay triangulation.