using the Modify Trace Appearance dialog. For further discussion, see **Overriding the Color of Contour Traces** on page II-373.

Contour Trace Names

The name of a contour trace is usually something like “zwave=2.5”, indicating that the trace is the contour of the z data set “zwave” at the z=2.5 level. A name like this must be enclosed in single quotes when used in a command:

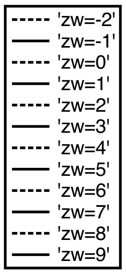
```
ModifyGraph mode('zwave=2.5')=2
```

This trace naming convention is the default, but you can create a different naming convention when you first append a contour to a graph using the /F flag with AppendMatrixContour or AppendXYZContour. The contour dialogs do not generate /F flags and therefore create contours with default names. See the **AppendMatrixContour** operation on page V-33 and the **AppendXYZContour** operation on page V-39 for a more thorough discussion of /F.

Example: Contour Legend with Numeric Trace Names

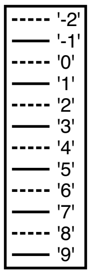
To make a legend contain slightly nicer wording you can omit the “zwave=” portion from trace names with `/F="%g"`:

```
AppendMatrixContour/F="%g" zw    // trace names become just numbers
```



```
----- 'zw=-2'
───── 'zw=-1'
----- 'zw=0'
───── 'zw=1'
----- 'zw=2'
───── 'zw=3'
----- 'zw=4'
───── 'zw=5'
----- 'zw=6'
───── 'zw=7'
----- 'zw=8'
───── 'zw=9'
```

`/F="%s=%g"` (default)



```
----- '-2'
───── '-1'
----- '0'
───── '1'
----- '2'
───── '3'
----- '4'
───── '5'
----- '6'
───── '7'
----- '8'
───── '9'
```

`/F="%g"`

You can manually edit the legend text to remove the single quotes around the trace names. Double-click the legend to bring up the Modify Annotation dialog.

For details on creating contour plot legends, see **Contour Legends** on page II-377.

Contour Trace Programming Notes

The **TraceNameList** function returns a string which contains a list of the names of contour traces in a graph. You can use the name of the trace to retrieve a wave reference for the private contour wave using **Trace-NameToWaveRef**. See also **Extracting Contour Trace Data** on page II-376.

If a graph happens to contain two traces with the same name, “instance notation” uniquely identifies them. For example, two traces named “2.5” would show up in the Modify Trace Appearance dialog as “2.5” and “2.5#1”. On the command line, you would use something like:

```
ModifyGraph mode('2.5'#1)=2
```

Notice how the instance notation (the “#1” part) is outside the single quotes. This instance notation is needed when the graph contains two contour plots that generate identically named traces, usually when they use the same /F parameters and draw a contour line at the same level (z=2.5).

See **Instance Notation** on page IV-20. Also see **Contour Instance Names** on page II-376.

The Color of Contour Traces

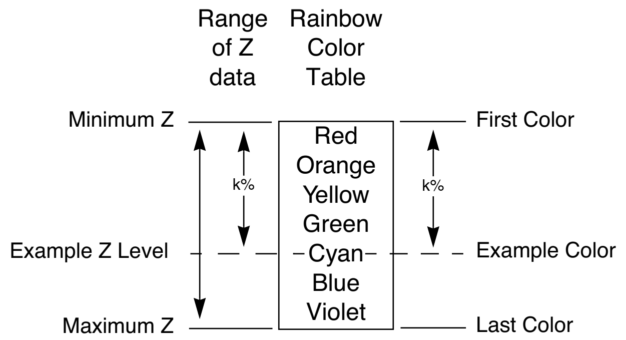
By default, contour traces are assigned a color from the Rainbow color table based on the contour trace’s Z level. You can choose different colors in the Contour Line Colors subdialog of the Modify Contour Appear-ance dialog.

The Contour Line Colors dialog provides four choices for setting the contour trace colors:

1. All contour traces can be set to the same color.
2. The color for a trace can be selected from a "color index wave" that you created, by matching the trace's Z level with the color index wave X index.
3. The color for a trace can be computed from a chosen built-in color table or from a color table wave that you created, by matching the trace's Z level with the range of available colors.

Color Tables

When you use a color table to supply colors, Igor maps the Z level to an entry in the color table. By default, Igor linearly maps the entire range of Z levels to the range of colors in the table by assigning the minimum Z value of the contour data to the first color in the table, and the maximum Z value to the last color:



Automatic Mode Color Table Mapping

With the Modify Contour Appearance dialog, you can assign a specific Z value to the color table's first and last colors. For example, you can use the first half of the color table by leaving First Color on Auto, and typing a larger number (2*(ZMax -ZMin), to be precise) for the Last Color.

To see what the built-in color tables look like, see **Color Table Details** on page II-395.

You can create your own color table waves that act like custom color tables. See **Color Table Waves** on page II-399 for details.

Color Index Wave

You can create your own range of colors by creating a color index wave. The wave must be a 2D wave with three columns containing red, green, and blue values that range from 0 (zero intensity) to 65535 (full intensity), and a row for each color. Igor finds the color for a particular Z level by choosing the row in the color index wave whose X index most closely matches the Z level.

To choose the row, Igor converts the Z level into a row number as if executing:

```
colorIndexWaveRow= x2pnt(colorIndexWave,Z)
```

which rounds to the nearest row and limits the result to the rows in the color index wave.

When the color index wave has default X scaling (the X index is equal to row number), then row 0 contains the color for z=0, row 1 contains the color for z=1, etc. By setting the X scaling of the wave (Change Wave Scaling dialog), you can control how Igor maps Z level to color. This is similar to setting the First Color and Last Color values for a color table.

Color Index Wave Programming Notes

Looking up a color in a color index wave can be expressed programmatically as:

```
red=   colorIndexWave (Z level)[0]
green= colorIndexWave (Z level)[1]
blue=  colorIndexWave (Z level)[2]
```

where ( ) indexes into rows using X scaling, and [ ] selects a column using Y point number (column number).