CHAPTER

14

The GCONTOUR Procedure

Overview

The GCONTOUR procedure produces plots that represent three-dimensional relationships in two dimensions. Lines or areas in a contour plot represent levels of magnitude (z) corresponding to a position (x, y) on a plane.

Use the GCONTOUR procedure to

- examine trends in your data when they contain too many peaks and valleys to be accurately observed using the G3D procedure
- examine data in which the levels, not the shape, of the data are important.
- identify contour levels using lines or patterns.

Note: The GCONTOUR procedure produces rectangular contour plots, not irregular contour maps. △

About Contour Plots

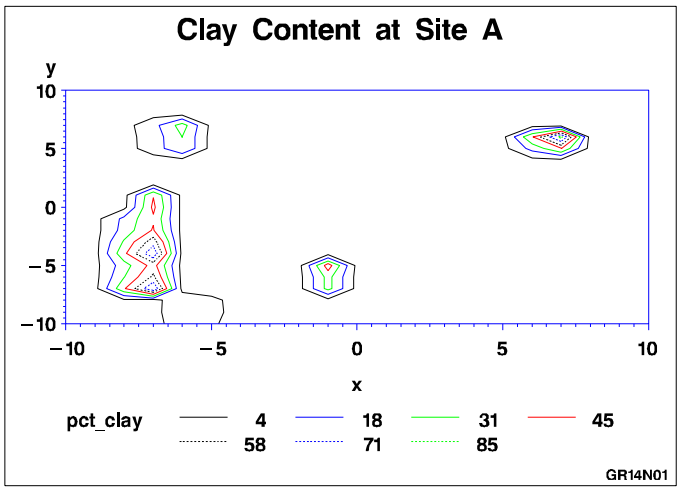
Contour plots represent the levels of magnitude of a variable z, called the *contour variable*, for a position on a plane given by the values of two variables x and y. Contour lines of different colors and line types show different levels of magnitude of z for locations of x and y.

Figure 14.1 on page 626 shows a simple contour plot that illustrates the percentage of clay found in soil samples at various locations of a testing site. The x and y axes on

the plot represent a graph of surface height at various *x*-*y* locations. The contour lines within the plot represent the locations on the plane that have the clay percentages specified in the legend. The program for this plot is in Example 1 on page 641.

By default, the GCONTOUR procedure automatically scales the axes to include the maximum and minimum data values, labels each axis with the name of its variable or an associated label, and draws a frame around the plot. In addition, it plots values using seven contour levels of the contour variable, representing those levels with default colors and line types. Finally, it generates a legend that is labeled with the contour variable's name.

Figure 14.1 Sample Contour Plot (GR14N01)

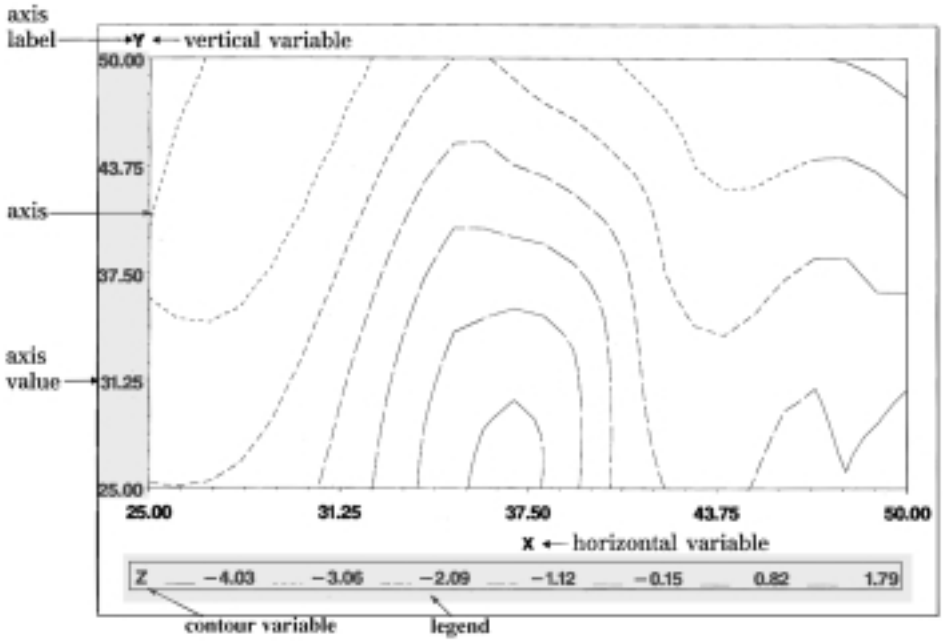


Concepts

Parts of a Contour Plot

Some terms used in the discussion of the GCONTOUR procedure are illustrated in Figure 14.2 on page 627.

Figure 14.2 GCONTOUR Procedure Terms



About the Input Data Set

The GCONTOUR procedure requires data sets that include three numeric variables: *x* and *y* for the horizontal and vertical axes, respectively, and *z* for the contour level. The observations in the input data set should form a rectangular grid of *x* and *y* values and exactly one *z* value for each *x*, *y* combination. For example, data that contain 5 distinct values of *x* and 10 distinct values for *y* should be part of a data set that contains 50 observations with values for *x*, *y*, and *z*. If a single *x*, *y* grid location has more than one associated *z* value, only the last such observation is used.

Interpolating Additional Values

Data sets often contain so few combinations of *x*, *y*, and *z* values that the GCONTOUR procedure cannot produce a contour plot. By default, the data set must contain nonmissing *z* values for at least 50 percent of the grid in order for the GCONTOUR procedure to produce a satisfactory plot. If your data are clustered in relatively small patches over a larger study area, you can use the PROC GCONTOUR statement's INCOMPLETE option, which allows plotting of data when over 50 percent of the plot grid contains missing data.

When the GCONTOUR procedure cannot produce a satisfactory contour plot because of missing *x*, *y*, or *z* values, SAS/GRAPH software issues an error message, and no graph is produced. To correct this problem, you can use the G3GRID procedure to process data sets to be used by the GCONTOUR procedure. The G3GRID procedure interpolates the necessary values to produce a data set with nonmissing *z* values for every combination of the *x* and *y* variables. The G3GRID procedure can also smooth data for use with the GCONTOUR procedure. You can use the output data set from the G3GRID procedure as the input data set for the GCONTOUR procedure. For an example of using PROC G3GRID to interpolate values, see Example 1 on page 641.

Procedure Syntax

Requirements: At least one PLOT statement is required.

Global statements: AXIS, FOOTNOTE, LEGEND, PATTERN, SYMBOL, TITLE

Reminder: The procedure can include the BY, FORMAT, LABEL, NOTE, and WHERE statements.

Supports: Output Delivery System (ODS)

PROC GCONTOUR <DATA=*input-data-set*>
<ANNOTATE=*Annotate-data-set*>
<GOUT=<*libref.*>*output-catalog*>
<INCOMPLETE>;
PLOT *plot-request* </*option(s)*>;

PROC GCONTOUR Statement

Identifies the data set that contains the plot variables. Optionally specifies annotation and an output catalog.

Requirements: An input data set is required.

Syntax

PROC GCONTOUR <DATA=*input-data-set*>
<ANNOTATE=*Annotate-data-set*>
<GOUT=<*libref.*>*output-catalog*>
<INCOMPLETE>;

Options

ANNOTATE=*Annotate-data-set*
ANNO=*Annotate-data-set*
specifies a data set to annotate all graphs produced by the GCONTOUR procedure. To annotate individual graphs, use ANNOTATE= in the action statement.

See also: Chapter 10, "The Annotate Data Set," on page 403

DATA=*input-data-set*
specifies the SAS data set that contains the variables to plot. By default, the procedure uses the most recently created SAS data set.

See also: "SAS Data Sets" on page 25 and "About the Input Data Set" on page 627

GOUT=< *libref.* >*output-catalog*
specifies the SAS catalog in which to save the graphics output produced by the GCONTOUR procedure. If you omit the libref, SAS/GRAPH looks for the catalog in the temporary library called WORK and creates the catalog if it does not exist.

See also: "Creating and Specifying Catalogs" on page 50

INCOMPLETE
allows plotting of data when over 50 percent of the plot grid contains missing data.

PLOT Statement

Creates contour plots using values of three numeric variables from the input data set as the source of the contour coordinates.

Requirements: A plot request is required.

Global statements: AXIS, FOOTNOTE, LEGEND, PATTERN, SYMBOL, TITLE

Description The PLOT statement specifies the three variables to plot. Optionally, it controls the contour levels, labels the plot lines, and modifies axes as well as the general appearance of the graph. Only one plot request can be specified in a PLOT statement. To specify multiple plots for a single PROC GCONTOUR statement, use multiple PLOT statements.

The PLOT statement automatically

- plots the values using seven contour levels of the *z* variable
- scales the axes to include the maximum and minimum data values
- labels the *x* and *y* axes and displays the contour levels in the plot's legend
- draws a frame around the plot.

Note: You cannot produce overlaid contour plots with PROC GCONTOUR alone. △

You can use global statements to modify the axes, the legend, the contour lines and contour line labels, and the fill patterns and pattern colors for contour areas. You can also add titles, footnotes, and notes to the chart, and you can use an Annotate data set to enhance the chart.

Syntax

PLOT *plot-request* </*option(s)*>;

plot-request must be

*y***x*=*z*

option(s) can be one or more options from any or all of the following categories:

- appearance options:

  ANNOTATE=*Annotate-data-set*
  CAXIS=*axis-color*
  CFRAME=*background-color*
  COUTLINE=*outline-color*

CTEXT=*text-color*
GRID
NOAXIS | NOAXES
NOFRAME

- horizontal axis options:
  AUTOHREF
  CHREF=*reference-line-color*
  HAXIS=AXIS<1...99>
  HMINOR=*number-of-minor-ticks*
  HREF=*value-list*
  HREVERSE
  LHREF=*line-type*
  XTICKNUM=*number-of-ticks*
- vertical axis options:
  AUTOVREF
  CVREF=*reference-line-color*
  LVREF=*line-type*
  VAXIS=AXIS<1...99>
  VMINOR=*number-of-minor-ticks*
  VREF=*value-list*
  VREVERSE
  YTICKNUM=*number-of-ticks*
- contour options:
  CLEVELS=*color(s)*
  JOIN
  LEGEND=LEGEND<1...99>
  LEVELS=*value-list*
  LLEVELS=*line-type-list*
  NLEVELS=*number-of-levels*
  NOLEGEND
  PATTERN
- labeling options:
  AUTOLABEL | AUTOLABEL=(*autolabel-suboptions*)
  where *autolabel-suboptions* can be one or more of these:
  CHECK=*checking-factor* | NONE
  MAXHIDE=*amount*<*units*>
  REVEAL
  TOLANGLE=*angle*
- catalog entry description options:
  DESCRIPTION='*entry-description*'
  NAME='*entry-name*'

Required Arguments

***y**x=z**
specifies three numeric variables from the input data set:

y	is the variable that is plotted on the vertical (*y*) axis.
x	is the variable that is plotted on the horizontal (*x*) axis.
z	is the variable that is plotted as contour lines.

Options

You can specify as many PLOT options as you want, and you can list them in any order. If you use a BY statement on the procedure, the options in each PLOT statement affect all graphs produced by that BY statement.

ANNOTATE=*Annotate-data-set*
ANNO=*Annotate-data-set*
specifies a data set to annotate charts produced by the PLOT statement.

See also: Chapter 10, "The Annotate Data Set," on page 403

AUTOHREF
draws reference lines at all major tick marks on the horizontal axis.

AUTOLABEL | AUTOLABEL=(*autolabel_suboptions*)
automatically labels the contour lines. *Autolabel-suboptions* can be one or more of the suboptions described in "Autolabel Suboptions" on page 636.

The label for each contour line is the *z* value for that contour level. By default, labels are displayed in BEST format. To change the format, use a FORMAT statement.

When AUTOLABEL is used, the LLEVELS= and CLEVELS= options are ignored and the SYMBOL statement controls label text and contour-line attributes. For more information on the SYMBOL statement, see "Modifying Contour Lines and Labels with the SYMBOL Statement" on page 640.

Even though AUTOLABEL labels the contour lines, a default legend is still generated. To suppress the legend, use the NOLEGEND option.

Featured in: Example 2 on page 643

AUTOVREF
draws reference lines at all major tick marks on the vertical axis.

CAXIS=*axis-color*
specifies a color for axis lines and all major and minor tick marks. By default, axes are displayed in the second color in the colors list.

If you use the CAXIS= option, it may be overridden by the COLOR= suboption of the MAJOR= or MINOR= option in an AXIS definition.

CFRAME=*background-color*
CFR=*background-color*
fills the axis area with the specified color and automatically draws a frame around the axis area.

CHREF=*reference-line-color*
CH=*reference-line-color*
specifies the color for reference lines that are requested by the HREF= option. By default, these lines are displayed in the axis color.

CLEVELS=*color(s)*
specifies a list of colors for plot contour levels. The number of specified colors should correspond to the number of contour levels since one color represents each level of contour. If fewer colors are specified than the number of levels in the plot, the

procedure provides default colors from the current colors list. The procedure default is to rotate through the current colors list for each line type.

This option is ignored if AUTOLABEL is used.

COUTLINE=*outline-color*

specifies a color for outlining filled areas. This option is ignored unless the PATTERN option is also used. By default, the outline color is the same as the color of the filled area.

Note: The outline color is the only distinction between empty patterns. Use of this option makes the patterns look the same when VALUE=EMPTY in PATTERN definitions. △

Featured in: Example 4 on page 647

CTEXT=*text-color*

specifies a color for all text on the axes and legend, including axis labels, tick mark values, legend labels, and legend value descriptions.

If you omit the CTEXT= option, a color specification is searched for in this order:

1. specified colors on assigned AXIS and LEGEND statements
2. the CTEXT= option in a GOPTIONS statement
3. the default, the first color in the colors list.

For legend text, colors that you specify on an assigned LEGEND statement override CTEXT=. Thus, a LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

For axes text, colors that you explicitly specify for values and labels on an assigned AXIS statement override CTEXT=. Thus, an AXIS statement's VALUE= color is used for axis values, and its LABEL= color is used for axis labels. However, if the AXIS statement specifies only general axis colors with its COLOR= option, the CTEXT= color overrides that general specification and is used for axis labels and values; the COLOR= color is still used for all other axis colors, such as tick marks.

If you use a BY statement in the procedure, the color of the BY variable labels is controlled by the CBY= option in the GOPTIONS statement.

Featured in: Example 4 on page 647

CVREF=*reference-line-color*
CV=*reference-line-color*

specifies the color for reference lines that are requested by the VREF= option. By default, these lines are displayed in the axis color.

DESCRIPTION='*entry-description*'
DES='*entry-description*'

specifies the description of the catalog entry for the chart. The maximum length for *entry-description* is 40 characters. The description does not appear on the chart. By default, the GCONTOUR procedure assigns a description of the form PLOT OF *y***x*=*z*, where *y***x*=*z* is the request specified in the PLOT statement.

GRID

draws reference lines at all major tick marks on both axes. This is the same as specifying both AUTOHREF and AUTOVREF.

HAXIS=AXIS<1...99>

assigns axis characteristics from the corresponding AXIS definition to the horizontal (*x*) axis.

See also: "AXIS Statement" on page 162

Featured in: Example 2 on page 643

HMINOR=*number-of-minor-ticks*
HM=*number-of-minor-ticks*
specifies the number of minor tick marks to use between each major tick mark on the horizontal (*x*) axis. No values are displayed for minor tick marks. HMINOR= overrides the MINOR= option in an AXIS definition assigned to the horizontal (*x*) axis.

HREF=*value-list*
draws one or more reference lines perpendicular to the horizontal axis at points specified by *value-list*. See the LEVELS= option on page 633 for a description of *value-list*.

HREVERSE
specifies that the order of the values on the horizontal axis be reversed.

JOIN
combines adjacent grid cells with the same pattern to form a single pattern area. This option is ignored unless the PATTERN option is also used.

Featured in: Example 4 on page 647

LEGEND=LEGEND<1...99>
assigns legend characteristics to the legend to adjust the location, text, and appearance of axes on the plots. The option value indicates which LEGEND definition to use. To suppress the legend, use the NOLEGEND option. The LEGEND= option is ignored if the specified LEGEND definition is not currently in effect.

If you use the SHAPE= option in a LEGEND statement, the value LINE is valid. If you use the PATTERN option, SHAPE=BAR is also valid.

See also: "LEGEND Statement" on page 187

Featured in: Example 2 on page 643

LEVELS=*value-list*
specifies values of *z* for plot contour levels and therefore changes the number of contour levels. You can specify up to 100 values. By default, the GCONTOUR procedure plots seven contour levels for *z*. These levels occur at every 15th percent of the range between the 5th and 95th percentiles.

For numeric variables, *value-list* can be an explicit list of values, a starting and an ending value with an interval increment, or a combination of both forms:

- *n* <...*n*>
- *n* TO *n* <BY *increment*>
- *n* <...*n*> TO *n* <BY *increment* > <*n* <...*n*> >

If a numeric variable has an associated format, the specified values must be the *unformatted* values.

By default, the GCONTOUR procedure selects colors and line types for the contour levels by rotating through the colors list for each line type (1 through 46) until all the levels have been represented. The level lines on the plot represent the intersection of a plane, parallel to the *x-y* plane, and the surface that is formed by the data at the *z* value. See "Selecting Contour Levels" on page 636 for more information.

You can specify the colors and line types for contour levels. The way to do this depends on whether AUTOLABEL is used:

- If AUTOLABEL is used, the SYMBOL statement controls colors and line types for contour levels. See "Modifying Contour Lines and Labels with the SYMBOL Statement" on page 640 for more information.
- If AUTOLABEL is not used, the CLEVELS= and LLEVELS= options control colors and line types for contour levels.

As an alternative to representing contour levels with contour lines, you can use the PATTERN option to fill each level with a solid pattern or with the colors and patterns specified in PATTERN statements.

Featured in: Example 2 on page 643 and Example 3 on page 645

LHREF=*line-type*
LH=*line-type*
specifies the line type for drawing reference lines that are requested by the HREF= option. *Line-type* is a number from 1 to 46. The default is LHREF=1, a solid line. See "Specifying Line Types" on page 248 for available line types.

LLEVELS=*line-type-list*
lists numbers for line types for plot contour lines. Each line type represents one contour level, so the number of line types listed should correspond to the number of contour levels. Thus, for a contour plot that uses the default seven levels, specify seven line types.

If fewer line types are specified than the number of levels in the plot, the procedure provides default line types. With the default, contour levels rotate through line types 1 through 46, displaying each line type in all of the colors in the colors list before moving to the next line type. See "Specifying Line Types" on page 248 for available line types.

For colors and lines specified with both the CLEVELS= and LLEVELS= options, the first contour level is displayed in the first color in the CLEVELS= color list and in the first line type specified with the LLEVELS= option. The second level is displayed in the second color and the second line type, and so on.

This option is ignored if AUTOLABEL is used.

Featured in: Example 3 on page 645

LVREF=*line-type*
LV=*line-type*
specifies the line type for drawing reference lines that are requested by the VREF= option. *Line-type* is a number from 1 to 46. The default is LVREF=1, a solid line. See "Specifying Line Types" on page 248 for available line types.

NAME='*entry-name*'
specifies the name of the catalog entry for the graph. The maximum length for *entry-name* is 8 characters. The default name is GCONTOUR. If the specified name duplicates the name of an existing entry, SAS/GRAPH software adds a number to the duplicate name to create a unique entry, for example, GCONTOU1.

NLEVELS=*number-of-levels*
specifies the number of contour levels to plot. Values can be integers from 1 to 100, inclusive. The contour levels are computed as follows, where **L** represents an array of levels:

- If the value of NLEVELS= is less than 7, then

```
D = (Zmax - Zmin) / NLEVELS
d = 0.5 * D
L[0]  = Zmin + d, L[i]  = L[i-1]  + D
```

  In this case, each level is the midpoint of a number of ranges equal to the value of the NLEVELS= option. These ranges exactly cover the range of the *z* variable.

- If the value of NLEVELS= is greater than or equal to 7, then

```
e = 0.05 * (100 - NLEVELS) / 93
d = (Zmax - Zmin) * e
D = ((Zmax - Zmin - 2*d) / (NLEVELS - 1)
```

```
L[0]  = Zmin + d, L[i]  = L[i-1]  + D
```

In this case, the first and last midpoints are set closer to the minimum and maximum *z* values as the values of NLEVELS= gets closer to 100, and the remaining midpoints are equally spaced between them.

NOAXIS
NOAXES
specifies that a plot have no axes, axis values, or axis labels. The frame is displayed around the plot unless you use the NOFRAME option.

NOFRAME
suppresses the frame that is drawn by default around the plot area.

NOLEGEND
suppresses the plot legend that describes contour levels and their line types or fill patterns and colors.

PATTERN
specifies the fill pattern and pattern colors for contour areas. The plot contour levels are represented by rectangles filled with patterns. The pattern for each rectangle is determined by calculating the mean of the values of the *z* variable for the four corners of the rectangle and assigning the pattern for the level closest to the mean.

By default, the procedure uses a solid pattern for the levels and rotates the pattern through the colors list. If the V6COMP option is in effect for the GOPTIONS statement, cross-hatch patterns are used instead of solid patterns. To explicitly define patterns, use PATTERN definitions for map/plot patterns.

See also: "Selecting Contour Levels" on page 636

Featured in: Example 4 on page 647

VAXIS=AXIS<1...99>
assigns axis characteristics from the corresponding AXIS definition to the vertical (*y*) axis.

See also: "AXIS Statement" on page 162

Featured in: Example 2 on page 643

VMINOR=*number-of-minor-ticks*
VM=*number-of-minor-ticks*
specifies the number of minor tick marks located between each major tick mark on the vertical (*y*) axis. No values are displayed for minor tick marks. The VMINOR= option overrides the MINOR= option in an AXIS definition that is assigned to the vertical (*y*) axis.

VREF=*value-list*
draws one or more reference lines perpendicular to the vertical axis at points specified by *value-list*. See the LEVELS= option on page 633 for a description of *value-list*.

VREVERSE
specifies that the order of the values on the vertical axis be reversed.

XTICKNUM=*number-of-ticks*
YTICKNUM=*number-of-ticks*
specify the number of major tick marks located on a plot's *x* or *y* axis, respectively. The value of *n* must be 2 or greater. The defaults are XTICKNUM=5 and YTICKNUM=5.

The MAJOR= or ORDER= option in an AXIS definition that is assigned to the *x* axis overrides the XTICKNUM= option. The MAJOR= or ORDER= option in an AXIS definition that is assigned to the *y* axis overrides the YTICKNUM= option.

Autolabel Suboptions

In the AUTOLABEL= option, *autolabel-suboptions* can be one or more of the suboptions described here.

CHECK=*checking-factor* | NONE
specifies a collision checking factor that controls collisions between contour label text and other contour lines or other labels. Values can be integers from 0 to 100, inclusive, where 0 provides minimal collision checking and 100 provides maximal collision checking. Fractional values are permitted. The default is CHECK=75.

CHECK=NONE suppresses contour label collision checking and may substantially lessen the time needed to compute the contour graph.

Specifying a checking factor can slow the graph procedure. Generally, CPU time required to fit all the labels increases in direct proportion to the size of the CHECK= number.

MAXHIDE=*amount*<*units*>
specifies the maximum amount of contour line that can be hidden by contour labels. The value of *amount* must be greater than zero.

Valid *units* are CELLS (horizontal character cell positions), CM (centimeters), IN (inches), or PCT (percentage of the width of the graphics output area). The default is MAXHIDE=100PCT. If you omit *units*, a unit specification is searched for in this order:

1. the GUNIT= option in a GOPTIONS statement
2. the default unit, CELLS.

If you specify units of PCT or CELLS, the MAXHIDE= suboption calculates the amount of contour line that can be hidden based on the width of the graphics output area. For example, if you specify MAXHIDE=50PCT and if the graphics output area is 9 inches wide, the maximum amount of the contour line that can be hidden by labels is 4.5 inches.

This option maintains data integrity. It provides a check for overly small increments in the STEP= option in the SYMBOL statement. Additionally, it can prevent small contours from being significantly hidden even when the value of STEP= is sufficiently large.

REVEAL
specifies that the contour lines are visible through the label text as dashed lines. Line style 33 is used. This option provides a simple way to see all portions of labeled contours and can be used to inspect the label positions with respect to the contour lines. It is primarily used for debugging. Occasionally, single-character contour labels can be placed off center from the clipped portion of the contour line when the contour line is irregular or jagged.

TOLANGLE=*angle*
specifies the maximum angle (the tolerance angle) between any two adjacent characters of a contour label. The value of *angle* must be between 1 and 85 degrees, inclusive. The default is TOLANGLE=30. To force contour labels to fall on very smooth sections, specify a small tolerance angle.

Selecting Contour Levels

You can use the LEVELS= option to select the contour levels for your plot. You use LEVELS= values differently, depending on whether you specify the PATTERN option in the PLOT statement.

When you do not use the PATTERN option, the levels represent the intersection of a plane (parallel to the *x*-*y* plane at the *z* value) and the surface formed by the data. That

is, if you use the data to create a surface plot with the G3D procedure, the contour lines in a GCONTOUR procedure plot represent the intersection of the plane and the surface.

For example, suppose that you use the G3D procedure, and your data produces a surface plot like the one shown in Figure 14.3 on page 637. The same data used with this PLOT statement in the GCONTOUR procedure produces a contour plot like the one in Figure 14.4 on page 637:

```
plot y*x=z / levels=-7.5 to 7.5 by 2.5;
```

The contour lines in Figure 14.4 on page 637 represent the intersection of the surface in Figure 14.3 on page 637 with planes parallel to the plane formed by the variables *x* and *y* and located at *z* values of –7.5, –5.0, –2.5, and so on.

Figure 14.3 Surface Plot

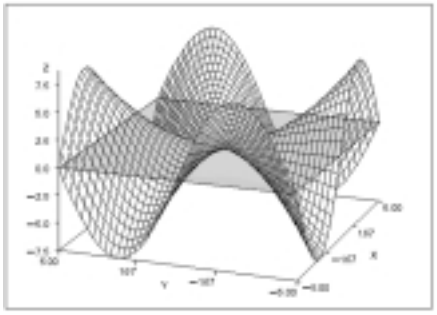
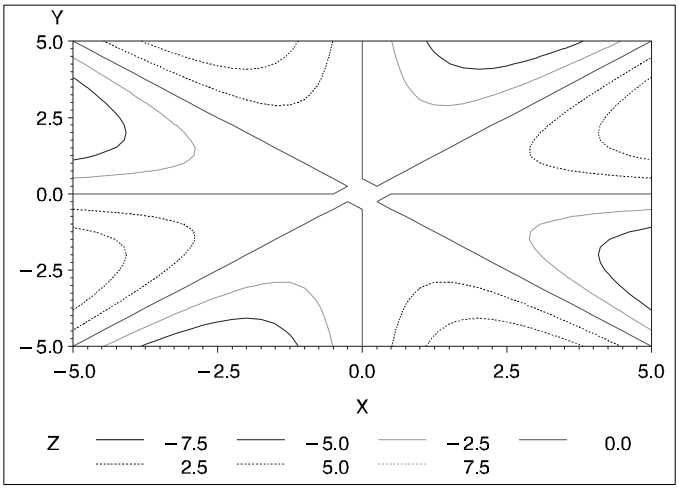


Figure 14.4 Line Contour Levels

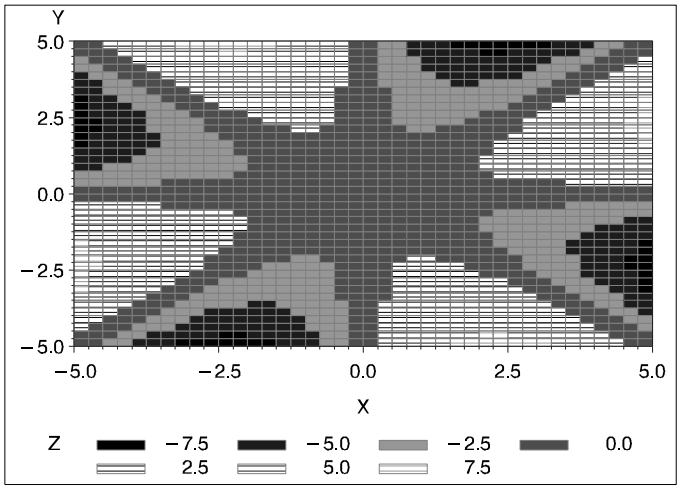


When you use the PATTERN option, contour levels are represented by rectangles filled with patterns. The rectangles are formed by points in the *x*-*y* grid. The contour pattern of a rectangle, or grid cell, is determined by the mean or average value of the *z* variable for the four corners of the rectangle. The grid cell is assigned the pattern for the level closest to the calculated mean. For example, if you have specified contour levels of 0, 5, and 10, and the plot contains a grid cell with a mean of 100, it is assigned

the pattern for the nearest level: 10. A grid cell with a mean of 7.6 will also be assigned the pattern for the 10 level.

Figure 14.5 on page 638 shows a contour plot with the PATTERN option that uses the same data and contour levels as Figure 14.4 on page 637. The pattern for the rectangle is assigned depending on the mean of the grid values at the four corners. As a result, two contour plots using the same contour levels can present your data differently if one plot uses a pattern and the other does not. The contour pattern boundaries do not correspond to the contour lines shown in Figure 14.4 on page 637.

Figure 14.5 Pattern Contour Levels



Specifying Axis Order

You can use AXIS statements to modify the text and appearance of plot axes, and then you can assign the axes to the contour plot with the PLOT statement's HAXIS= and VAXIS= options. If the AXIS statement uses an ORDER= option, there are special considerations for using that AXIS definition with the GCONTOUR procedure.

A list of variable values that are specified with the AXIS statement's ORDER= option must contain numbers listed in ascending or descending order; these numbers are treated as a continuous data range for an axis. Thus, for a contour line or pattern to span the entire specified range, it is not necessary for the maximum and minimum values of the list to match exactly with the maximum and minimum data values of the corresponding *x* or *y* variable. For example, suppose that you assign this AXIS definition to the horizontal (*x*) axis:

```
axis1 order=-2.5 to 2.5 by .5
```

Suppose also that the horizontal axis variable has these values: –5, –4, –3, –2, –1, 0, 1, 2, 3, 4, 5. Depending on the data, contours could extend through the full range of the ORDER= list rather than from –2 to 2, which are the actual values of the variable assigned to the horizontal (*x*) axis. In this case, values are interpolated for the *x* variable at any point where the *y* variable intersects the minimum axis value (–2.5) or the maximum axis value (2.5). Data values that are outside of the axis range (in this case, –5, –4, –3, 3, 4, and 5) are clipped from the plot.

When ORDER= lists cause data clipping, internal plotting grids are modified according to these rules:

- If an ORDER= list causes data clipping on a single axis, linear interpolation generates the *z* values of the starting and/or ending column of the plotting grid.

For example, in the previous example, the value of *z* is interpolated for –2.5 and 2.5 on the horizontal (*x*) axis.

- If ORDER= lists cause data clipping on both axes, the response variable values of the new corners are derived by fitting the new *x*, *y* location on a plane formed by three of the original four points of the corresponding grid square.

In addition, if you assign the following AXIS definition to a plot of the same data, the contour levels on the plot will not extend beyond the range of the data:

```
axis1 order=-10 to 10 by 1;
```

To see the effects of the ORDER= option:

- Figure 14.6 on page 639 shows the effects when the range of ORDER= values matches the range of values for the variables assigned to the horizontal (*x*) and vertical (*y*) axes.
- Figure 14.7 on page 640 shows the effects when the range of ORDER= values is smaller than the range of data values.
- Figure 14.8 on page 640 shows the effects when the range of ORDER= values is larger than the range of data values.

Figure 14.6 Effects of the AXIS Statement's ORDER= Option: ORDER= values match variable values

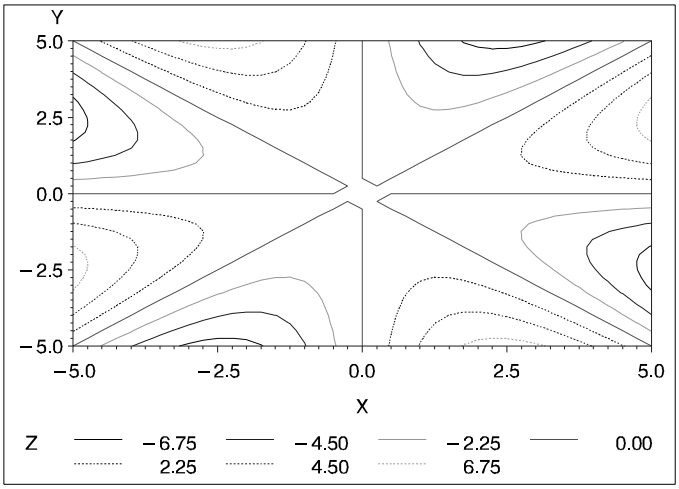


Figure 14.7 Effects of the AXIS Statement's ORDER= Option: ORDER= range is smaller than variable range

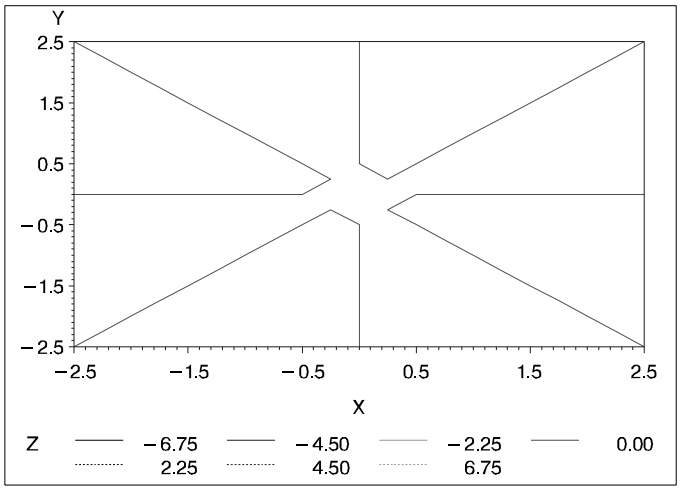
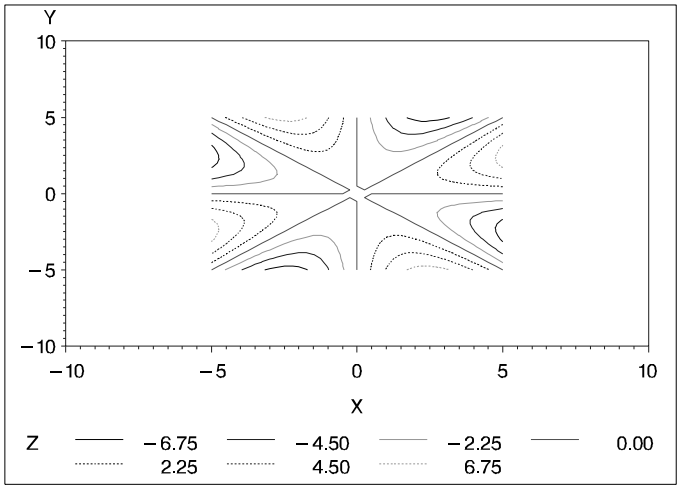


Figure 14.8 Effects of the AXIS Statement's ORDER= Option: ORDER= range is larger than variable range



Modifying Contour Lines and Labels with the SYMBOL Statement

When you use the AUTOLABEL option, the LLEVELS= and CLEVELS= options are ignored, and contour-line and label attributes are controlled by the SYMBOL statement. Defaults are used if not enough SYMBOL statements are specified to match the number of contour levels.

If a SYMBOL statement does not include a color option, that statement may be applied to more than one contour level. In this case, the SYMBOL statement is used once with every color in the colors list and generates more than one SYMBOL definition. See "SYMBOL Statement" on page 226 for details.

Table 14.1 on page 641 describes how SYMBOL statement options affect contour plot lines and labels.

Table 14.1 The Effect of SYMBOL Statement Options on Contour Lines and Labels

SYMBOL Statement Option	Contour Line or Label Element Affected
LINE=*line-type*	Contour line style
WIDTH=*n*	Contour line thickness
CI=*line-color* or COLOR=*color*	Contour line color
FONT=*font*	Contour label font
HEIGHT=*height*	Contour label height
CV=*color* or COLOR=*color*	Contour label color
STEP=*distance<units>*	Minimum distance between labels on the same contour line
VALUE='*text*'	Contour label text
VALUE=NONE	Suppresses the contour label text

The SYMBOL statement option INTERPOL= is not supported by the GCONTOUR procedure.

The STEP= option specifies the minimum distance between contour labels. The lower the value, the more labels the procedure uses. A STEP= value of less than 10 percent is ignored by the GCONTOUR procedure and a value of 10 percent is substituted.

See STEP= on page 240 for more information.

Specifying Text for Contour Labels To override the default labels that are displayed by the AUTOLABEL option, you can specify label text for one or more contour lines. To do so, use both the FONT= and VALUE= options on the SYMBOL statement that is assigned to the contour level. Default labels are used for contour levels that you do not label.

For example, this SYMBOL1 statement displays the text string **`Highest`** in the Swiss font on the contour line that it modifies:

```
symbol1 font=swiss value='Highest';
```

You must specify both FONT= and VALUE= or the text is not used. For an example, see Example 2 on page 643.

Examples

Example 1: Generating a Simple Contour Plot

Procedure features:

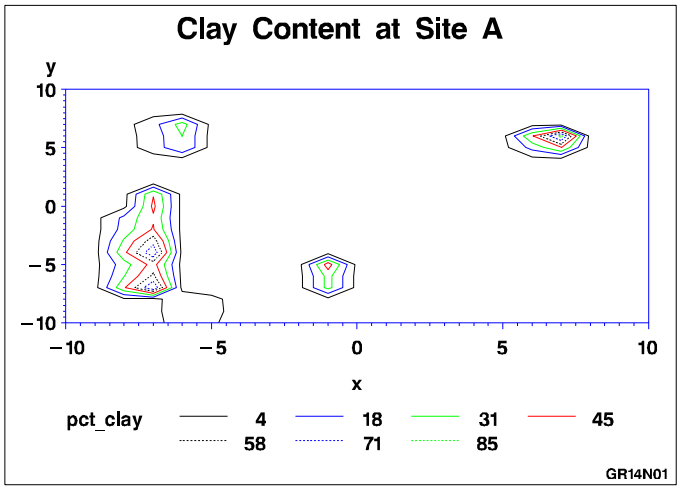
PLOT statement

Other features:

FORMAT statement

G3GRID procedure

Sample library member: GR14N01



This example shows a simple contour plot that describes the percentage of clay found in soil samples at various locations of a testing site. By default, the axes are scaled to include all data values and are labeled with the names of the axes variables. Values are plotted with seven contour levels, which are represented by contour lines with default colors and line types. The default contour levels occur at every 15th percent of the range between the contour variable's 5th and 95th percentile. The legend is labeled with the contour variable's name and identifies the contour levels that are included in the plot.

This example uses the G3GRID procedure to interpolate clay percentages for grid cells that do not have percentages in the data. Without the G3GRID procedure, there are too many missing values for the percentages, and the GCONTOUR procedure cannot produce a satisfactory contour plot.

Assign the libref and set the graphics environment.

```
libname reflib 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
         colors=(black blue green red)
         ftext=swiss ftitle=swissb htitle=6 htext=4;
```

Create the data set. REFLIB.CLAY contains the percent of clay at various locations of a test site.

```
data reflib.clay;
   input x y pct_clay;
   datalines;
-10    -10       2.316
-10     -9       1.816
-10     -8       2.427
...more data lines...
 10      8         .
 10      9         .
 10     10         .
;
```

Interpolate values for the contour plot. The interpolated data set is stored in REFLIB.CLAY2.

```
proc g3grid data=reflib.clay out=reflib.clay2;
   grid y*x=pct_clay / naxis1=21
                       naxis2=21
                       join;
run;
```

Define title and footnote.

```
title1 'Clay Content at Site A';
footnote1 j=r 'GR14N01 ';
```

Generate a simple contour plot. The procedure uses REFLIB.CLAY2, the output data set from PROC G3GRID. To simplify the legend labels, clay percentages are formatted with no decimal positions.

```
proc gcontour data=reflib.clay2;
   format pct_clay 2.0;
   plot y*x=pct_clay;
run;
quit;
```

Example 2: Labeling Contour Lines

Procedure features:

PLOT statement options:

AUTOLABEL=
HAXIS=
LEGEND=
LEVELS=
VAXIS=

Other features:

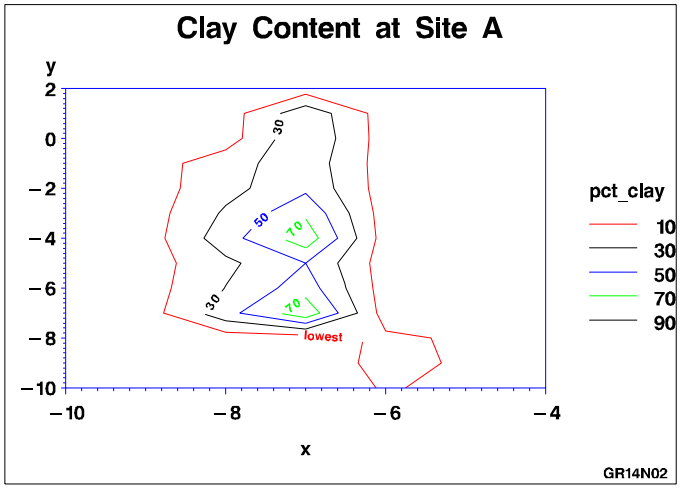
AXIS statement

LEGEND statement

SYMBOL statement

Data set: REFLIB.CLAY2 on page 643

Sample library member: GR14N02



This example modifies Example 1 on page 641 to label contour levels with the AUTOLABEL option. When AUTOLABEL is used, the SYMBOL statement controls the labels and attributes of contour lines. In this example, SYMBOL1 defines a text label for the lowest contour level. Each remaining contour line gets the default label, which is the contour variable's value at that contour level. All the contour lines are solid, which is the default line type for the SYMBOL statement.

This example also uses AXIS statements to limit the plot to one of the contour areas from the output of Example 1 on page 641, and it uses a LEGEND statement to move the legend so the procedure has more room for displaying the *y* axis.

Assign the libref and set the graphics environment.

```
libname reflib 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
         colors=(black blue green red)
         ftext=swiss ftitle=swissb htitle=6 htext=4;
```

Define title and footnote.

```
title1 'Clay Content at Site A';
footnote1 j=r 'GR14N02 ';
```

Define axes characteristics. AXIS1 uses ORDER= to set major tick marks at 2-unit intervals from -10 to -4. AXIS2 uses ORDER= to specify 2-unit intervals from -10 to 2. These axes ranges effectively zoom in on one of the contour areas from Example 1 on page 641.

```
axis1 order=(-10 to -4 by 2);
axis2 order=(-10 to 2 by 2);
```

Define legend characteristics. POSITION= centers the legend to the right of the graphics area, and LABEL= positions the legend label above the legend entries. ACROSS= places legend entries in rows 1 entry wide.

```
legend1 position=(right middle)
        label=(position=top)
```

```
        across=1;
```

Define symbol characteristics. SYMBOL1 specifies a font and text string to label the lowest-level contour lines. COLOR= ensures that each SYMBOL definition is used only once. In SYMBOL2, STEP= increases the number of contour labels by placing the labels closer together than the default distance of 65 percent.

```
symbol1 height=2.5
        font=swissb
        value='lowest'
        color=red;
symbol2 height=2.5
        step=25pct
        color=black;
symbol3 height=2.5
        color=blue;
symbol4 height=2.5
        color=green;
```

Generate the contour plot. LEVELS= specifies contour levels from 10 to 90 at 20-unit intervals. AUTOLABEL= turns on labeling, and CHECK=NONE turns off collision checking so the maximum number of contour labels can be displayed. HAXIS= and VAXIS= assign AXIS definitions to the plot. LEGEND= assigns the LEGEND1 definition to the plot.

```
proc gcontour data=reflib.clay2;
   plot y*x=pct_clay / levels=10 to 90 by 20
                       autolabel=(check=none)
                       haxis=axis1
                       vaxis=axis2
                       legend=legend1;
run;
quit;
```

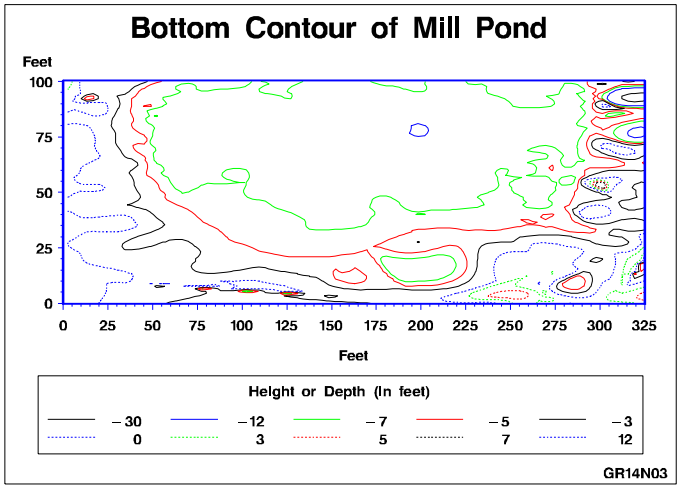
Example 3: Specifying Contour Levels

Procedure features:

PLOT statement options:

LEVELS=
LLEVELS=

Sample library member: GR14N03



This example generates a contour plot that shows the height or depth of a pond and its surrounding land. In the example, the PLOT statement uses the LEVELS= and LLEVELS= options to specify explicit contour levels and line types for the contour plot. It also uses a LEGEND statement to modify the plot's default legend.

This example uses the G3GRID procedure to interpolate points for grid cells that do not have a needed dimension in the data. Without the G3GRID procedure, there are too many missing values for the point locations, and the GCONTOUR procedure cannot produce a satisfactory contour plot.

Assign the libref and set the graphics environment.

```
libname reflib 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
         colors=(black blue green red)
         ftext=swiss ftitle=swissb htitle=6 htext=3;
```

Create the data set. REFLIB.POND contains the raw data for a pond floor and surrounding land.

```
data reflib.pond;
   input vdist hdist height;
   datalines;
10    88       0
18    55      -1
24    22.5    -1.67
...more data lines...
64    272.5  -6.25
60    277.5  -6.5
62    277.5  -6.5
;
```

Define title and footnote.

```
title 'Bottom Contour of Mill Pond';
footnote j=r 'GR14N03 ';
```

Define axis characteristics.

```
axis1 order=(0 to 325 by 25) width=3 minor=(n=4)
      label=('Feet');
axis2 order=(0 to 100 by 25) width=3 minor=(n=4)
      label=(' Feet');
```

Define legend characteristics.

```
legend1 frame shape=line(7)
        label=(position=top j=c 'Height or Depth (in feet)');
```

Interpolate points for the contour plot.

```
proc g3grid data=reflib.pond out=reflib.pondgrid;
   grid vdist*hdist=height / naxis1=100 naxis2=100;
run;
```

Generate the contour plot. LEVELS= specifies the values of the contour levels. LLEVELS= sets the line types for the contour lines. Solid lines identify negative contour levels, and dashed lines identify positive contour levels.

```
proc gcontour data=reflib.pondgrid;
   plot vdist*hdist=height /levels= -30 -12 -7  -3 0 3 5 7 12
                            llevels=  1   1  1  1  1 2 2 2 2  2
                            legend=legend1
                            haxis=axis1
                            vaxis=axis2;
run;
quit;
```

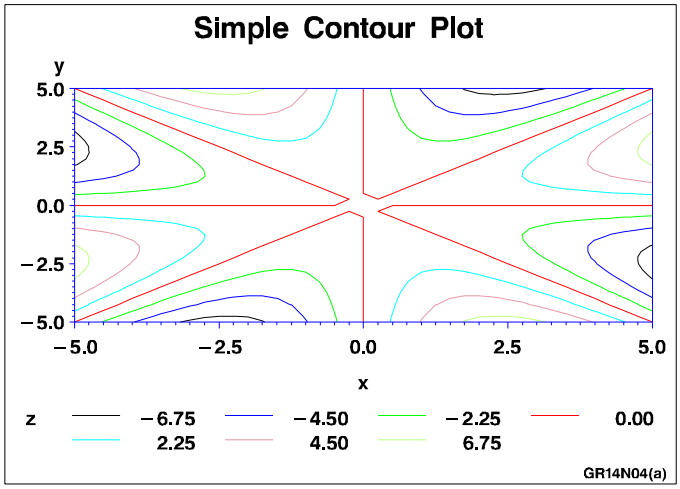
Example 4: Using Patterns and Joins

Procedure features:

PLOT statement options:

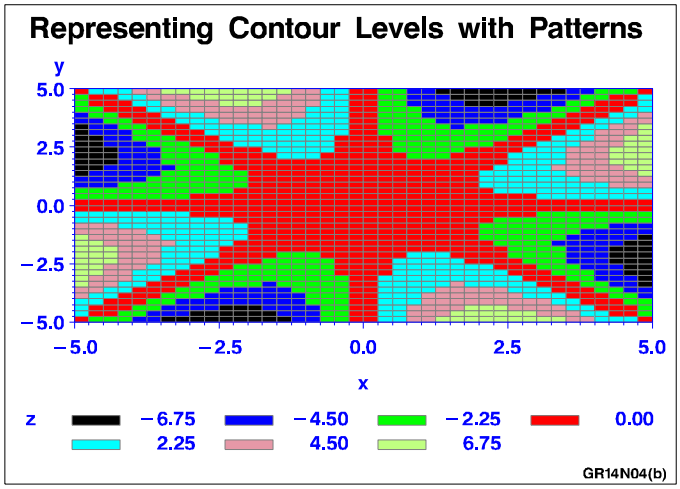
- COUTLINE=
- CTEXT=
- JOIN
- PATTERN

Sample library member: GR14N04

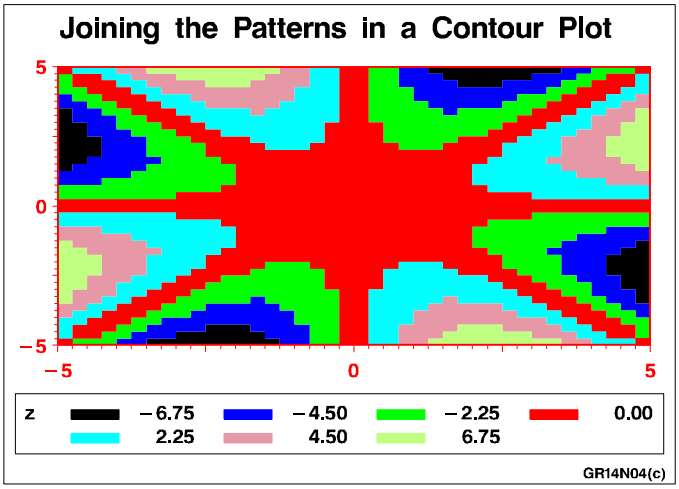


This example demonstrates the differences between using lines and patterns to represent contour levels. It first uses a simple PLOT statement to generate the default output, which uses lines to represent contour levels.

As shown in the following output, the example then modifies the PLOT statement by specifying the PATTERN option, which uses patterns to distinguish between contour levels. Additional PLOT statement options outline filled areas in gray and specify green text for all text on the axes and in the legend.



Finally, as shown by the following output, the example uses the JOIN option to combine the patterns in grid cells for the same contour level. Additional options enhance the plot by modifying the axes and framing the legend.



Assign the libref and set the graphics environment.

```
libname reflib 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
         colors=(black blue green red)
         ftext=swiss ftitle=swissb htitle=6 htext=4;
```

Create the data set. REFLIB.SWIRL is generated data that produces a symmetric contour pattern, which is useful for illustrating the PATTERN option.

```
data reflib.swirl;
   do x= -5 to 5 by 0.25;
      do y= -5 to 5 by 0.25;
         if x+y=0 then z=0;
         else z=(x*y)*((x*x-y*y)/(x*x+y*y));
         output;
      end;
   end;
run;
```

Define title and footnote for the default output.

```
title 'Simple Contour Plot';
footnote j=r 'GR14N04(a) ';
```

Generate a simple contour plot.

```
proc gcontour data=reflib.swirl;
   plot y*x=z;
run;
```

Define title and footnote for second plot.

```
title 'Representing Contour Levels with Patterns';
footnote j=r 'GR14N04(b) ';
```

Generate the contour plot. PATTERN fills the contour levels with solid patterns. COUTLINE= names the color that outlines the grid cells. CTEXT= names a color for axes and legend text.

```
proc gcontour data=reflib.swirl;
   plot y*x=z / pattern
                coutline=gray
                ctext=green;
run;
```

Define title and footnote for last plot.

```
title 'Joining the Patterns in a Contour Plot';
footnote j=r 'GR14N04(c) ';
```

Define axis and legend characteristics for last plot. Blanks are used to suppress tick labels at positions -2.5 and 2.5.

```
axis1 label=none value('' ' ' '0' ' ' '5')
      color=red width=3;
axis2 label=none value('' ' ' '0' ' ' '5')
      color=red width=3;

legend frame;
```

Generate the last contour plot. JOIN combines grid cells for the same contour levels.

```
proc gcontour data=reflib.swirl;
   plot y*x=z / pattern
                join
                haxis=axis1
                vaxis=axis2
                legend=legend1;
run;
quit;
```

References

Snyder, W.V. (1978), "Contour Plotting [J6] ," *ACM Transactions on Mathematical Software*, 4, 290–294.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/GRAPH® Software: Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS/GRAPH® Software: Reference, Version 8

ISBN 1-58025-525-6

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999