UNITED STATES DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

Plotting Azimuthal Stress Data on Standard Map Projections Using Geoplot

Mara E. Schiltz [1]

Open-File Report 86-389

1. U.S. Geological Survey, MS 977
345 Middlefield Rd.
Menlo Park, CA 94025

INTRODUCTION

Geoplot is a plotting package, designed to run on the USGS O.E.V.E. UNIX operating system and to plot on either the 4014 Tektronix system or on a Calcomp plotter (Ward, 1983). Geoplot was written to plot many types of geophysical data on graphs and on a number of standard map projections. Geoplot is a child process under UNIX that can be reached interactively through the Geolab command language (Herriott, 1980) or can be reached from a compiled FORTRAN or C program. This report describes using the Geolab command language to access Geoplot to plot azimuthal data points, each consisting of a latitude, longitude, and an angle measured clockwise from north, on standard map projections. Plotting of azimuthal data on a Mercator projection about an arbitrary pole is also described. This report supplements the Geoplot and Geolab manuals by providing further examples of writing Geoplot "operators" for plotting data in the many map projections supported by Geoplot. The report is intended for users who are familiar with the basics of UNIX, and who have read the tutorials in the Geoplot and Geolab manual. For reference, the standard map projections available for use with Geoplot are given in the Appendix.

FUNDAMENTALS OF GEOLAB AND GEOPLOT

The basis for this report is a presentation of the methods used to create the plots of azimuthal data shown in Figures 1, 2, and 3. To make the methods presented later in the report easier to understand, a review of the basic building block in Geolab, the "operator", is presented. Readers already familiar with Geolab may wish to skip to the next subsection where the more difficult concepts from the later examples are explained and illustrated.

All of the Geolab and Geoplot parameters and commands are called "operators". The Geolab operators are listed in Herriott (1980) and the Geoplot operators are listed in Ward (1983). The user can also define operators, building on the Geolab parameters (Herriott, 1980). For example, if the user wants to add 2 numbers interactively in

Geolab, type

4 + 2 is *is displays the result on the terminal*

Geolab responds *6*

(In the interactive Geolab work of this section the user input is indicated by bold type; italicized numbers indicate Geolab responses; the italicized strings are comments added for further clarification.)

To interactively install the above equation in a Geolab operator, type

op add (4 + 2 is)

Now type **add** to get as before the Geolab response *6*. Complex operators can be created by combining a number of simple operators. For example, if we define a second simple operator

op divide(4 / 2 is)

The two operators can be combined to form a new operator

op both(add divide)

The result of typing **both** is:

6
2

To generalize the operator "add", the ^ operator is used:

op add(^ = x ^ = y x+y is)

and is applied in the following manner:

add 4 2

Using the ^ operator, "add" assigns 4 to x and 2 to y and adds the two numbers as before.

Before looking at methods for creating Figures 1 through 3, it's useful to discuss a few general concepts. The first concept is the method of data storage in Geolab. The Geolab memory is similar to the stack geometry of an HP calculator memory (Herriott, 1980). When 3 numbers are input sequentially, the first number that was input ends up 3 places down in the stack, and the third number input ends up on the top of the stack. Using Geolab, 3 numbers can be pushed onto the stack by typing

6 7 8

The three values that were pushed onto the stack can be displayed by typing

```
isn 3
```

The Geolab response is

```
8
7
6
```

The usual way to achieve the same result would be to type

```
8=x   7=y   6=z
```

but the less familiar method of stacked-memory variable assignment will be used in the more complex examples presented later in this report.

"Pushing" and "popping" the stack are useful in assigning values of the stack to variables to be used in calculations. For example, to assign values on the stack to x,y,z in Geolab, integer variables are declared:

```
intg x   intg y   intg z
```

The same three values as before are pushed onto the stack and then assigned to z, y, x respectively by typing

```
=x ,=y ,=z
```

Here, the top value on the stack, 8 is assigned to x; the value 8 is popped off the stack using the "," operator (Herriott, 1980), then the top value on the stack, 7, is assigned to y; the value 7 is popped off the stack with the "," operator; the top, and last value on the stack, 6, is stored in z.

Now, the values of x,y, z can be displayed by typing

```
x is  y is  z is
```

Geolab responds:

```
8
7
6
```

Do Loops and Indexing in One Dimensional Arrays

Two additional important concepts utilized in all three examples are do loops and indexing in one-dimensional arrays. The form of a do loop in Geolab is

```
number_of_iterations do (command)
```

For example,

6 do ("hello" is) *character strings are enclosed in quotes in Geolab*

will cause "hello" to be written on the screen six times. A useful operator to generalize the number of iterations is **len** which equals the length of an array. Given an array "data" of six elements, the following command sequence

len data do ("hello" is)

gives the same result as before.

To simplify indexing in the Geoplot operators that follow, one-dimensional arrays are used. The data for these examples is stored in a UNIX C shell file called "stress-data" (Figure 5). The data is translated into a one-dimensional array in Geolab, by reading the file across each row from left to right creating one long string of numbers. For example, the first two lines of the UNIX C shell file "stressdata" (Figure 5) are:

```
¯87.910 31.650 305.000
¯135.767 57.012 40.000
```

A data file consisting of these two lines could be read into a Geolab eight element, one-dimensional array, called "data", by using a user-defined operator "getdata" which contains four commands that generalize the method given for input/output on p. 7 of the Geolab tutorial and uses a standard FORTRAN format:

```
op getdata( data open ^ =input   read data "4f8.3"   close input)
```

The fourth field will contain zeroes when the data file is read into Geolab and will be used for temporary storage of the output of calculations (see subsection "Calculating true local azimuths"). Now, after typing

real data:8 *variable declaration of the array*
getdata "stressdata" data is

Geolab responds:

```
¯87.91 31.65 305.00 0.0 ¯135.77 57.01 40.00 0.0
```

The elements of "data" can be generalized using an index "z":

data=[data:z data:(z+1) data:(z+2) data:(z+3) ... data:(z+8)]

Now, the incrementing equation, **4i-3=z with i=1 .. 3** can be used to treat the elements of the one-dimensional array, "data", in sets of four, where each set consists of the longitude(data:z), latitude(data:(z+1)) and azimuth (data:(z+2)) and the zero field (data:(z+3)) of a single stress data point. Utilizing this type of generalization, the user can write Geoplot operators that contain do loops to make calculations and to plot large one-dimensional arrays (see subsection "Do loops and indexing in one-dimensional arrays").

Data Space Versus Page Space

Converting latitudes, longitudes, and azimuths of the stress data from data space coordinates to page space coordinates is the key to the method used to plot the stress data using Geoplot. Data space refers to the coordinate system in which the data is defined. For a map the location of a point in data space is its latitude and longitude. Page space is the whole physical page of the plotting device measured in page units. The page space location of a point is its distance, measured in page units, from the x and y axes of the page with the origin in the lower left-hand corner. For the Tektronix 4014 the dimensions of the page are 1.31 page units in the x or horizontal direction by 1.0 page units in the y or vertical direction, where 10 inches=1 page unit. For the Calcomp plotter, the dimensions are 1 page unit in the y direction, and up to 7 page units in the x or rolling direction, where 33 inches (width of paper) = 1 page unit.

Conversion from a data space location to a page space location can be done by using a "movdrw" command (Ward, 1983, p.36) of the form:

```
x y mode movdrw
```

According to Geoplot convention, x corresponds to longitude, and y corresponds to latitude, a mode of -5 translates data space coordinates to page space coordinates.

Example: -100. 40. -5 movdrw isn 3 *isn 3 displays the top 3 values of the stack*

Geoplot responds

```
-5
.347    latitudinal distance from y axis described in page units
.491    longitudinal distance from x axis described in page units
```

Symbol-plotting Geoplot operators use page space angles, not data space angles, to plot the data. This distinction is important because map projections are, in general, not rectilinear (latitude and longitude curves are perpendicular); hence, for non-rectilinear projections, corrections need to be made to represent the input azimuths accurately with respect to the latitude/longitude lines of the particular projection. To understand how to make these corrections, it is necessary to distinguish between data space locations and angles versus those of page space as will be presented in the later subsection "Calculating true local azimuths".

Plotting the data

The Geoplot operator **letter** is used to plot the stress data. The form is:

x y -1 string angle 1 **letter**

The instructions to the Geoplot operator **letter** are: the point (x, y) is given, where x corresponds to longitude (data:z), and y corresponds to latitude (data:(z+1)); the mode "-1" is given which tells Geoplot to move to the data point x,y rather than a page space point. Any character string can be used as a plotting symbol and must be enclosed in quotes. The user can also specify an angle (orientation) for the symbol to be plotted at. Geoplot convention is that this angle is measured counter-clockwise from the horizontal axis of a standard Cartesian coordinate system.

We apply this to the stress data by defining an operator "plotdata":

op plotdata(**len** data/4 **do**(4*i-3=z, data:z data:(z+1) -1 "|" -data:(z+2) 1 **letter**))

The do loop contained in "plotdata", increments through the data set the number of times defined by the length of the array "data" divided by 4 ("len data/4"). The equation, 4*i-3=z is used to index each data point, consisting of longitude, latitude, azimuth, and a zero field. (The Geolab integer register "i" is an implicit index, which in this case has a range of 1 to "len data/4".) The "," operator pops the value of z off the stack and then the letter operator discussed previously is generalized to plot the data.

Azimuth plotting symbols

Character string combinations that can be used to plot the stress data are unlimited. The symbol used to plot the data in Figures 1, 2, and 3 is a result of plotting the data twice using the operator "plotdata" defined previously. The vertical bar "|" was plotted first at a user-specified size and then "\<" was plotted at a smaller size to result in an image of mirrored, inward-pointing arrows centered on the data point (Figure 1). A character string that consists of a backslash plus a character with no spaces between the two plots as the character string plus its mirror image centered about a point.

The size of a character string plotting symbol is determined by the Geoplot operator **letheight**. Letter height is based on the measurement of page units which was already discussed. The default value of letheight is .011 page units.

The easiest way to plot the two symbols at the same point to create a composite symbol such as the inward-pointing arrows is to further generalize "plotdata" by declaring a character-string variable "symbl" to which we can assign any combination of characters. For example:

```
char symbl:10
op plotdata(len data/4 do(4*i-3=z data:z data:(z+1) -1 symbl -(data:(z+2)) 1
        letter))
```

This could be implemented for a result such as that of the arrows in figure 1 by typing the following:

```
"|"=symbl    .015=letheight    plotdata
"\<"= symbl   .005=letheight   plotdata
```

Other composite symbols can be constructed using the backspace operator, "\b". For example, an asterisk enclosed by an oval could be plotted by specifying "*\b O" as the character string in the user-defined operator, "plotdata", from the previous subsection.

Specifying the angle at which a character string is plotted can be somewhat tricky. A character string is at 0° BEFORE it is rotated. (Recall that Geoplot angles are measured counter-clockwise from the horizontal axis.) For example, the vertical bar "|" is at 0° in Geoplot; rotated 90°, it would be a horizontal bar. In the method for plotting stress data presented in this paper, corrections for the angle of the vertical bar are made in two places. In the user-defined operator "plotdata" one correction is made for the fact that the Geoplot convention is to measure angles counter-clockwise in contrast to clockwise measurement of azimuth. Figure 6 shows a comparison between the convention of the azimuthal stress data and that of Geoplot. The correction is made by simply specifying the negative of the angle for the string to be plotted at in the operator "plotdata". A second correction required for correcting the plotted azimuth on non-conformal map projections is presented in step 3 of the following subsection. This step takes into account that angles in Geoplot are measured counter-clockwise from 0° on the x-axis of a Cartesian coordinate system in contrast to the azimuthal data which is measured clockwise from north (Figure 6). Because the symbol, "|", used to plot the data is at 0° before being rotated, the data space angle of the azimuths is $90° - azimuth$. The azimuths are read into Geolab as they are in the data file, but the translation is taken into account in the calculations of step 3 by utilizing the trigonometric law $cos(90° - az) = sin(az)$.

Calculating True Local Azimuths

Because the Geoplot symbol-plotting operators use the page space angles, we must establish a method for converting a data space angle to a page space angle. The azimuths (Figure 4) of the stress data which are measured clockwise from north must first be converted to the Geoplot convention for angles. To find the page space angle on most map projections, we must correct for the fact that the meridians are curved, and hence do not run vertically and horizontally on the page. This correction is made ~~trigonometrically, utilizing the fact that angles are preserved~~ locally in a Mercator projection (Snyder,1982). For example, to plot a 30° course measured clockwise from north on a Mercator projection, a line of that angle is drawn through the point started at. Thus, for a data point in any map projection, the user can translate that point to the Mercator projection and trigonometrically calculate another point an incremental distance away in the direction of the azimuth. These two points, when converted back to the desired map projection, describe the azimuth of the data point. Then, both points can be converted to page space, and the page space angle that the two points describe can be calculated using the arctangent function. The resulting page space angle, instead of the azimuths is then used to plot the stress data.

The following sequence of steps outlines the strategy for transforming the data space angles of the input data to page space angles for plotting in Geoplot via a Transverse Mercator projection.

Step 1:
```
14=trans
[ ¯140. ¯55. 25. 80. ]=datasp  4000000=mapscale
```

In the first step we set up the plot parameters. The operator, **trans**, specifies the map projection (see Appendix I) as the Mercator projection for initial calculations, though the actual plotted map will be a Transverse Mercator; **mapscale** specifies a scale of 1:40,000,000 **datasp** specifies the map boundaries where the form is [*low_longitude high_longitude low_latitude high_latitude*].

Step 2: data:z data:(z+1) -5 movdrw , =s: (z+1) , =s:z

The second step converts the longitude, data:z, and the latitude, data:(z+1), to page space coordinates and stores the values in the matrix "s" as s:z and s:(z+1).

```
Step 3:     cos((data:(z+2))* pi/180.) * .001 + s:(z+1) = s:(z+3)
            sin((data:(z+2))* pi/180.) * .001 + s:z = s:(z+2)
```

In the third step page space point is calculated at a distance of .001 page units (or any small number) from the point calculated in step 2 in the direction of the azimuth, data:(z+2), and stores it as (s:(z+2),s:(z+3)), after converting the input azimuth from degrees to radians for use with the trigonometric functions. Recall that the trigonometric law $cos(90^\circ - az) = sin(az)$ is used in this step.

Step 4: s:(z+2) s:(z+3) 5 movdrw , =data:(z+3) , =data:(z+2)

The fourth step converts the page space point s:(z+2), s:(z+3) back to data space, using the **movdrw** mode of 5 and stores the resulting data space values in data:(z+2) and data:(z+3). After this calculation "data" consists of 2 points, (data:z, data:(z+1)) and (data:(z+2), data:(z+3)), that are a small distance apart and define a line oriented in the direction of the given azimuth.

Step 5: 16 = trans ¯100=transcon:1

The fifth step switches the transformation to Transverse Mercator. The Geoplot operator **transcon** specifies the central meridian of 100° West (see Ward, 1983 for a detailed chart of which members of the **transcon** array should be changed for each map projection).

```
Step 6:     data:z data:(z+1) ¯5 movdrw , =y1 , =x1
            data:(z+2) data:(z+3) ¯5 movdrw , =y2 , =x2
            (atan((x2-x1)/(y2-y1)))/rad = data:(z+2)
```

In the first two lines of the sixth step the data space coordinates of the two points (data:z, data:(z+1) and data:(z+2)), data:(z+3)) on the Transverse Mercator projection are transformed to page space coordinates and are stored in the dummy points (x1, y1) and (x2, y2). The last line of step 6 uses the arctangent function to calculate the page space angle defined by the 2 points, and converts the calculated angle from radians to degrees, and stores it in data:(z+2).

This example transforms an azimuth for a single stress data point from a data space angle to a page space angle. To correct the entire data set, each step must be driven by a do loop as in the file of operators, "stressops", given in the next section.

PLOTTING DATA ON MAP PROJECTIONS

Now that we've discussed the general principles involved in plotting azimuthal data using Geoplot, we can look at the specific variations used to create the plots in Figures 1, 2, 3. Several options for methods of generating a plot are discussed in the Geoplot tutorial (Ward, 1983). When first starting with Geoplot, you'll probably want to experiment on the Tektronix 4014 by defining operators and giving commands interactively as we have done previously. But, once you are ready to generate plot segments for use on the Calcomp, the easiest method is to write a file in the UNIX text editor of operator definitions and read the file into Geolab by using the "lure" command of the form:

lure "filename"

Example 1

The following is an example of a UNIX C shell file "stressops" file of Geolab and Geoplot operators. The operators set up some of the parameters in Geolab when they are read into Geolab, but most of the file consists of user-defined operators that will cut down errors and typing time in the terminal session listing that follows. The file "stressops" was used to create Figure 1, a plot of the azimuthal data (Zoback, Zoback, and Schiltz, 1984) on a Transverse Mercator projection with a central meridian of 100 ° west. Most of the operators in "stressops" have been discussed previously. To provide further explanation italicized comments, set off by " %" have been added.

stressops

```
% operator to set transformation to Transverse Mercator projection
op TR(16=trans)
%operator to set data boundaries
op bounds([ ¯140. ¯50. 15. 80.]=datasp)
% operator to set transformation to Mercator projection
op TR1(14=trans)
%operator to set the mapscale
op scale(40000000=mapscale)
%operator to set the first element of transcon array to value of user-specified central meridian
op cmerid( ¯100=transcon:1)
%combines 3 previously listed operators
op getset(TR1 bounds cmerid)

%Fortran-style declarations of variables and variable arrays. Comments are added to indicate what
%is stored in the variable or variable array.
char input:50                 %name of the UNIX C shell file
char symbl:10                 %character string to be plotted
real data:(4*405)             %array containing the stress data
real s:(4*405)                %Dummy array for calculations
real del 0.001=del            %incremental distance for calculations
real rad   pi/180.=rad        %degrees-to-radians conversion factor
real x real y real x1 real y1 real x2 real y2  %dummy variables
real olon:20 real olat:20     %meridians and parallels to be drawn for reference—
                              number is generalized

%sets up necessary precision, for stress applications, 3 digits to the right of the decimal point
3=prec

% reads the stress data from a C shell file into the Geolab array "data"
op getdata(data open  ^ =input   read data "4f8.3"  close input)

% The following is the set of commands listed in steps 1 through 6 discussed in the previous
%subsection "Calculating Local Azimuths" with do loops added to drive the calculations
op adapt(len data/4 do (4*i-3=z ad1))   % incrementing operator for ad1
op ad1(data:z data:(z+1) ¯5 movdrw ,=s:(z+1) ,=s:z)
op transmute(len data/4 do(4*i-3=z mute1))   %incrementing operator for mute1
op mute1(cos(data:(z+2)*rad)*del +s:(z+1) =s:(z+3),
            sin(data:(z+2)*rad)*del +s:z =s:(z+2))
op reconvert(len data/4 do(i*4-3=z reconv1))   %incrementing operator for reconv1
op reconv1(s:(z+2) s:(z+3) 5 movdrw ,=data:(z+3) ,=data:(z+2))
op getangle(len data/4 do(4*i-3=z, getang1)))   %incrementing operator for getang1
op getang1(data:z  data:(z+1)  ¯5 movdrw ,=y1 ,=x1,  data:(z+2) data:(z+3) ¯5 movdrw ,=y2 ,=x2
            (atan((x2-x1)/(y2-y1)))/rad=data:(z+2))
```

```
%Combines the previous operators to do all the calculations
op all(scale getset adapt transmute reconvert bounds TR scale getangle)

%Plots the data
op plotdata(len data/4 do(4*i-3=z data:z data:(z+1) ¯1 symbl ¯(data:(z+2)) 1 letter))

%Longitude(olon) meridians and latitude(olat) parallels to be drawn.
[ ¯120. ¯100. ¯80. ¯60. ]= olon
[ 20. 40. 60.]=olat

%Operators to draw the latitude and longitude meridians:
%the first two operators are do loops to drive the second two which draw the meridians
~~op drwlon (len olon do(drawlon olon:i))~~
op drwlat (len olat do(drawlat olat:i))
op drawlon(^ =x, x datasp:4 -1 movdrw x datasp:3 -2 movdrw)
op drawlat(^ =y, datasp:1 y -1 movdrw datasp:2 y -2 movdrw)

%The following two operators specify the character string, "symbl", to be used
%in "plotdata" and specify the letter size to be used with that string. Letheight is
%measured in page units where the default value is .011 page units.
op short ( "\<"=symbl 0.005=letheight)
op long( "|"=symbl .015=letheight)
```

Now we can use the operators defined in the file "stressops" in the terminal session below to create a plot file called "plot.PL". The **hardops**, which are standard Geolab and Geoplot operators, are in bold face print; if you need more information about a hardop, refer to the Geolab and Geoplot manuals. If an operator is not in bold face, it was defined above and you should refer back to "stressops" for further information. The strings in quotes are UNIX C shell filenames. The comments are in italics. Note that most of the work consisted of defining the operators, so the listing of the terminal session is quite short.

This terminal session could be used on the tek type by specifying "tek" rather than "calcomp" as the "ploton" device. When using the tek tube, do not specify scale because Geoplot automatically scales your map to maximize its extent on the terminal screen. When a scale is specified that would result in a map too large for the page space of the device, Geoplot adjusts your data space values (the map boundaries) so that only a small portion of the desired map may appear on the Tek screen.

Terminal Session 1

gl	*enters Geolab and creates geolab.w in home directory*
lure "stressops"	*reads "stressops" into Geolab*
getdata "stress"	*reads the data file "stress" into Geolab*
ploton calcomp	*starts Geoplot and creates geoplot.w and plot.PL file*
all	*sets up the plot space and does the corrections (see "stressops")*
.007=letheight	*set the letter height for numbers on the frame*
frame	*draws a frame that windows the area defined by* **datasp**
short plotdata long plotdata	*specifies symbol,size of symbol and plots the data*
drwlat drwlon	*draws the latitude and longitude meridians*
e "outlines"	*causes Geolab to execute the UNIX C shell file "outlines" that windows the world coastline file, /usr/db/world.co.m (see appendix)*
"namer" **map**	*plots the mapfile "namer" created by "outlines"*
q	*quits Geolab and Geoplot, leaving the user in the UNIX C shell*

Now that we've looked at a detailed example, let's discuss what is stored in the UNIX C shell files "geoplot.w" and "geolab.w". When you give commands in Geoplot to set up the plot space such as specifying the map projection or the scale, you are actually setting values in the 257-member **gpcom** array which determines all the features of the plot space. These values are stored in the file "geoplot.w". The plot.PL file, created when you specify **calcomp** as the **ploton** device contains the machine instructions for generating the Calcomp plot. To use the Calcomp plotter from UNIX, set up the plotter and give the **calcomp plot.PL** command from the UNIX C shell (see a plotting expert at your facility for more information on using the Calcomp plotter).

If you make a mistake in within a Geoplot terminal session such as specifying the scale or dataspace incorrectly, there are two ways of resetting Geoplot. The first is to return to Geolab by typing

gpq

Then use ~~the **e** operator to execute the~~ C ~~shell command that removes~~ "geoplot.w" ~~and~~ "plot.PL":

e "rm geoplot.w plot.PL"

The second method is to reset the Geoplot common (**gpcom** array) to its default values using the **R** option of the **ploton** command. The syntax of this command is

ploton deviceR

Device can be replaced with "calcomp" or "tek". Resetting Geoplot is recommended because most of the elements of the **gpcom** array are interdependent. For example, changing the map transformation (**trans**) not only will change the mapscale and possibly the dataspace, but could also change any of the other elements of the **gpcom** array, probably producing undesired results. Needless to say, it is simpler to clear Geoplot than to search through a 257-member array for the offending element!

The "geolab.w" file contains all of the user-defined operators, including all user-defined arrays. If a mistake is made in Geoplot such as those discussed above, it is not necessary to clear Geolab. Operators can be erased, using the **erase op_name** command. Operators can be overwritten by using the **!** operator. For a single operator the **!op** declaration is used. An entire set of operators can be overwritten by using the **!lure** command. For example, to overwrite the operators from the file "stressops" with the operators from a newer version of the same file, type **!lure** "stressops"

Example 2

To display the same stress data on another projection with a new center point, we need only change a few of the values in the file "stressops" in the UNIX editor. Parameters that must be changed include: **trans** to change the projection; **transcon** to set a center point; **mapscale** to change the scale; **datasp** to change the map boundaries; user-defined operators "olat" and "olon" to change the latitude and longitude meridians drawn; dimensions of the user-defined arrays "data" and "s" if size of the data set is changed. For this example we will use an Orthographic projection with a center point of 50° North, 65°West. A copy of "stressops" should be made in the UNIX C shell and the appropriate operator definitions replaced with the following in the UNIX editor (or the commands can be preceeded with the ! operator from within Geolab):

```
op TR (21=trans)                           Specify orthographic projection
op cmerid( ¯65=transcon:1 50= transcon:2)  Specify center point
op bounds([ ¯150 ¯30 0 90]=datasp)         Set map boundaries
op scale(50000000.=mapscale)               specify the scale
[ 10. 20. 30. 40. 50. 60. 70. 80. ]=olat   specify latitudes for drawlat
[ -140. -130. -120. -110. -100. -90. -80. -70. -60. -50. -40. ]=olon
```

Figure 2 was created using the same commands as those of Terminal Session 1 except for the addition of another command to reduce the size of the rough map digitization, /usr/maps/world.co. In figure 1 we used the fine digitization that, because it is stored in 10 degree bands, must be windowed band by band using the **maputil** shell file "outlines" (appendix), whereas the rough digitization can be pared down with one **maputil** command. In both cases the same output map filename, "namer", was used as the ouput map filename. The commands to window and to plot the rough digitization are listed below. The Geolab command, **e**, incorporated below, allows the user to execute UNIX C shell commands from within Geolab. The command must be enclosed in double quotes.

Terminal Session 2 — addendum to map commands of Terminal Session 1

```
e "maputil -r ¯150. ¯30. 0. 90. -o namer /usr/maps/world.co"
"namer" map
```

Example 3

To create a Mercator projection about an arbitrary pole (eg. the pole of absolute motion of North America, transformation 30 is used. The arbitrary pole is specified and data points are transformed to new latitude/longitude coordinates relative to that pole. Maps of this type are generated in the same way as Figures 1 and 2: a file of operators, "ROTATEDOPS" (below), is created in the UNIX editor. This file is identical to "stressops" (Example 1) except for re-definitions of the plot spaces as in Example 2 and four new commands, and it is used in the subsequent terminal session as in the previous examples. The additions to "ROTATEDOPS" are the user-defined operators P1, bounds2, gr, and rotate. P1 is an operator that specifies the projection pole, in the example below, the Minster-Jordan (1978) pole of absolute motion for the North American plate, 76.05° west, 49.02° north has been used. This pole is ignored unless transformation 30 is in use, so it can be set at any time in Geoplot. To transform a data point to its location relative to a new pole, after specifying the transformation and pole, use the **movdrw** command as follows:

```
30=trans        [-76.05 49.02]= pole
-125. 32.5 -5 movdrw isn 3
```

Geoplot responds:

```
-5
50.25
275.90
```

The operator "bounds2" describes the data space array for the rotated space. Setting the boundaries for the plot of rotated data on the Mercator projection takes some manipulation because the rotated coordinates of the original data space generally rotate to coordinates defining a polygon oblique to the page. For example, the original data space for Figure 3, [⁻125. ⁻114. 32.5 42.], translated to the 4 corners: (262.84, 55.61), (270.45, 62.8), (286.1, 57.33), (275.9, 50.25). So a best guess at an enclosing box was made by choosing maximum and minimum longitude and latitude values, and then testing the shape of the rotated data space on Tektronix 4014. Through trial and error the proper latitude and longitude values that enclose all the data, [260. 287. 54. 63.], were determined.

The operator "rotate" is used to transform data points relative to a specified pole. To transform a point of stress data relative to a pole, the sequence of steps is the same as in the previous subsection "Calculating true local azimuths" where the operations of the "rotate" operator replace step 5. At the end of step 4 the result is our original data point and a point that is a small distance away in the direction of the azimuth on a Mercator projection. The following operator "rotate" changes the **trans** value switches to transformation 30, the rotated transformation; increments by twos throught the set of 4 fields; calls "rotatel", and returns to the Mercator projection. The operator "rotatel" translates the coordinates of each point to its new position relative to the Minster-Jordan (1978) pole of rotation for North America and stores the result in the current point.

```
op rotate (30=trans, len data/2 do(2*i-1=z rotatel) 14=trans)
op rotatel(data:z data:(z+1) -5 movdrw ,=data:(z+1) ,=data:z)
```

With the sets of transformed points that result from using these operators we can calculate the transformed azimuth using the same method as that presented in step 6 of the subsection "Calculating true local azimuths".

One of the important operators for your Geoplot toolbox is gr:

```
op gr ([ 0 2 0 1]= pictsp=gridsp=subjsp)
```

This operator resets the plotspace for use on the Calcomp plotter enlarging it by one page unit (33 inches) in the x-direction (direction of the change in longitude). The operator solves two problems. The first, which it was created to solve, is creating a map that extends more than one page unit in the x-direction. If the plotspace is not extended, Geoplot will adjust the dataspace and in some cases clip the frame also. The second, which is its purpose here, is to solve Geoplot's mistakes in adjusting the plottable space. In this example the rotated plot of California and Nevada at a scale of 1:6,000,000 (Figure 3) has the approximate dimensions of 11 by 8 inches. The operator "gr" used in the operator "all" (see ROTATEDOPS below) solves the problem of clipping of the map boundaries that result from the excessive shrinking of the gridspace and subject space and the paring down of the data space by Geoplot. (To check if the user-specified dataspace was adjusted by Geoplot, use the Geoplot operator **pspaces** after using the command **frame.**)

ROTATEDOPS

```
%operator to set transformation to Mercator projection
op TR (14=trans)
%operator to set data boundaries
%op bounds([¯125. ¯114. 32.5 42.]=datasp)
%Rotated boundaries of frame to enclose California and Nevada determined by trial and error
op bounds2([ 260. 287. 54. 63.]=datasp)
% operator to set pole to Minster and Jordan(1978) pole of rotation for North America
op P1([ -76.05 49.02]=pole)
%combines 3 previously listed operators
op getset(TR bounds P1)
%operator to set the mapscale
op scale(6000000=mapscale)
%operator to expand the plotspace to prevent clipping
op gr ([ 0 2 0 1]= pictsp=gridsp=subjsp)

char input:50
char symbl:10
real s:(240)
real data:(240)
real del 0.001=del
real rad   pi/180.=rad
real x real y real x1 real y1 real x2 real y2
real olon:2   real olat:2
3=prec
% latitude and longitude lines to be drawn to create a frame
[ 260. 287.]=olon
[ 54. 63.]=olat
%operator to read the stress data into Geolab
op getdata(data open  ^ =input  read data "4f8.3"  close input)

% Sequence of operators (called by "all") that transforms the stress data
%relative to a pole — see discussion above
op adapt(len data/4 do (4*i-3=z ad1))  %incrementing operator for ad1
op ad1(data:z data:(z+1) ¯5 movdrw ,=s:(z+1) ,=s:z)
op transmute(len data/4 do (4*i-3=z mute1))
op mute1(cos(data:(z+2)*rad)*del +s:(z+1) =s:(z+3),
      sin(data:(z+2)*rad)*del + s:z =s:(z+2))
op reconvert(len data/4 do(i*4-3=z reconv1))
op reconv1(s:(z+2) s:(z+3) 5 movdrw ,=data:(z+3) ,=data:(z+2))
% note that "rotate" increments by twos rather than fours as in the
%other operators of this sequence because the points resulting from
%the previous calculations are a series of points that are not related
%until the calculations of "getangle"
op rotate (30=trans, len data/2 do(2*i-1=z rotate1) 14=trans)
op rotate1(data:z data:(z+1) -5 movdrw ,=data:(z+1) ,=data:z)
op getangle(len data/4 do(4*i-3=z, getangl)))
op getangl  (  data:z data:(z+1) -5 movdrw ,=y1 ,=x1,
               data:(z+2) data:(z+3) -5 movdrw ,=y2 ,=x2 ,
               (atan((x2-x1)/(y2-y1)))/rad=data:(z+2))

op plotdata(len data/4 do(4*i-3=z data:z data:(z+1) -1 symbl (data:(z+2)) 1 letter))
op all(scale getset adapt transmute reconvert bounds2 scale rotate getangle)

% Operators to draw the latitude (olat) and longitude (olon) meridians:
%The first two operators are do loops the second two draw the meridians
op drwlon (len olon do(drawlon olon:i))
op drwlat (len olat do(drawlat olat:i))
op drawlon(^ =x, x datasp:4 -1 movdrw x datasp:3 -2 movdrw)
op drawlat(^ =y, datasp:1 y -1 movdrw datasp:2 y -2 movdrw)
op short (0.005=letheight "\ <"=symbl)
op long(.015=letheight "|"=symbl)
```

~~The third terminal session is almost identical to the first because the user-defined~~, complex operator "all" from "ROTATEDOPS" includes the operators that were added (getset, P1, bounds2, gr, rotate). The two changes in the terminal session are the methods of drawing the frame and of creating the map file. In Figure 3, **movdrw** was used to create a box rather than **frame**, because as can be seen from the translation of data space relative to the new pole (above), the longitude values are consistent but not related to the north pole or the pole of rotation. At present, the software to rotate the original latitude/longitude meridians does not exist, so those meridians are not included in Figure 3. The second change is that the continental outlines are rotated using **maputil** as shown in the terminal session below. This file was used to generate Figure 3, a rotation of the stress data (Zoback and Zoback, 1980) and the state outlines of California and Nevada about the Minster-Jordan (1978) pole of absolute motion of the North American plate, 76.05° west, 49.02° north.

Terminal Session 3

```
gl                            enter Geolab
lure "ROTATEDOPS"             read "ROTATEDOPS" into Geolab
getdata "california"          reads the data file "california" into Geolab
ploton calcomp                enter Geoplot
all                           map parameters set (including pole) and data rotated. See
                              introduction to Example 3 and ROTATEDOPS for more detail.
drwlat drwlon                 draw latitude and longitude box for frame
short plotdata long plotdata  plot the data
e "maputil -p -76.05 49.02 -r -125. -114. 32.5 42. -o calmap"
                              window data from "usmap" (see appendix) and rotate it about a
                              pole. Output file is "calmap"
"calmap" map                  draw the map created in preceding step
```

SUMMARY

With the two files of operators, stressops and rotatedops, the user should be able to create maps of oriented data on most standard map projections at any scale with only minor modifications. Once the user understands how to write Geoplot operators, the ways to symbolize data are unlimited. In addition, Geoplot has more map projections on which to present data than most graphic packages offer. Thus, the benefits of using Geoplot far outweigh any initial difficulties that the user might experience.

ACKNOWLEDGEMENTS

The drafts of this report were reviewed by Mary Lou Zoback, who made many helpful suggestions for improvement. Peter Ward not only reviewed the final draft and assisted me in overcoming many of the first technical difficulties I experienced in working with Geoplot, but gave me the initial ideas on how to approach this problem. Their assistance is greatly appreciated.

APPENDIX

Map Projections

The map projections available in Geoplot are:

Option No.	Projection
1.	Equirectangular
14.	Mercator conformal cylindrical
15.	Miller cylindrical
16.	Transverse conformal mercator cylindrical
17.	Universal transverse mercator
18.	Lambert azimuthal equal area
19.	Azimuthal equal-distance
20.	Gnomonic
21.	Orthographic azimuthal
22.	Perspective azimuthal
23.	Stereographic conformal azimuthal
24.	Lambert conformal conic
25.	Ptolemy equal interval conic
26.	Kavraiskiy IV equal interval conic
27.	Albers equal-area conic
28.	Polyconic
29.	Sinusoidal or Mercator equal area
30.	Rotated space–as mentioned in example 3, this is not actually a map projection

The projection is specified by the **n=trans** command, where **n** is the option number. The characteristics of each map transformation and the constants used in each projection, specified by the **transcon** values are summarized on p. 56–58 of the Geoplot manual. For a description of map projections, see Snyder (1982).

Digitizations

There are two world coastline digitizations on line on the PDP 11/70 UNIX system– /usr/db/world.co and /usr/db/world.co.m. Both are stored as binary files. The rough digitization of 5200 points, /usr/db/world.co, is displayed in Figure 2. Figure 1 shows the continental outlines of North America from /usr/db/world.co.m, a much more detailed digitization of the world coastlines consisting of 80,700 points. This file is stored in /usr/db in 10° latitude bands. For example, world.co.30 contains the 10° band from 30° North to 40°North.

The **-r** option of the **maputil** command is executed in the UNIX C shell, as shown in Terminal Session 2, to restrict the region of the map data to shorten plotting time and the resultant binary C shell map file is used in Geoplot with the **map** command. The following UNIX C shell file "outlines" was used to window the data in /usr/maps/world.co.m for Figure 1.

outlines

```
#Unix C shell file to window Geoplot map file, /usr/maps/world.co.m: each line windows a 10 degree
# band of data, storing it in an outfile. The last line compiles the data, using UNIX command
# "cat", into a file called "namer".
maputil -r -140. -40. 25. 80. -o out20 /usr/maps/world.co.20
maputil -r -140. -40. 25. 80. -o out30 /usr/maps/world.co.30
maputil -r -140. -40. 25. 80. -o out40 /usr/maps/world.co.40
maputil -r -140. -40. 25. 80. -o out50 /usr/maps/world.co.50
~~maputil -r -140. -40. 25. 80. -o out60 /usr/maps/world.co.60~~
maputil -r -140. -40. 25. 80. -o out70 /usr/maps/world.co.70
maputil -r -140. -40. 25. 80. -o out80 /usr/maps/world.co.80
cat out* > namer
```

The final digitization displayed in this report is an abbreviated version of the National Oceanographic Atmospheric Association digitization, which consists of the United States coastlines and conterminous state boundaries (Figure 3). Bob Simpson (USGS) created the abbreviated digitization by writing a FORTRAN program that deletes all points closer than .015 inches at a scale of 1:10,000,000 in the NOAA data set. The ASCII UNIX C shell file, written in free format with longitudes first((-)West) and latitudes second((+)North) was then converted to a binary file using the **-ai** option of the **maputil** command. The binary file was used with the Geoplot **map** command.

REFERENCES

Herriott, J., 1983, Geolab Programmer's Manual *in* ed., Meador, P.J., Programmer's manual for the UNIX timesharing system and the Geolab interactive environment, volume 7, section 1.

Herriott, J., 1983, A tutorial introduction to Geolab *in* ed., Meador, P.J., Programmer's manual for the UNIX timesharing system and the Geolab interactive environment. volume 7, section 1a.

Minster, J.B., Jordan, T.H., 1978, Present day plate motions, Journal of Geophysical Research, volume 83, p. 5331–5354.

NOAA, 1977, World Data Bank II, Volume 1, number PB-271 868. (Can be obtained from National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield. VA 22161)

Snyder, J.P., Map projections used by the U.S. Geological Survey, U.S. Geological Survey Bulletin 1532, 313 pages.

Zoback, M.L, Zoback, M.D., and Schiltz, M.E., 1984, Index of stress data for the North American and parts of the Pacific Plate, U.S. Geological Survey Open-File Report 84-157, 62 pages.

Ward. P.L., 1983. Geoplot–An advanced graphics package for mini-computers. part 1: User's guide, U.S. Geological Survey Open File report 83-807, 77 pages.

Figures

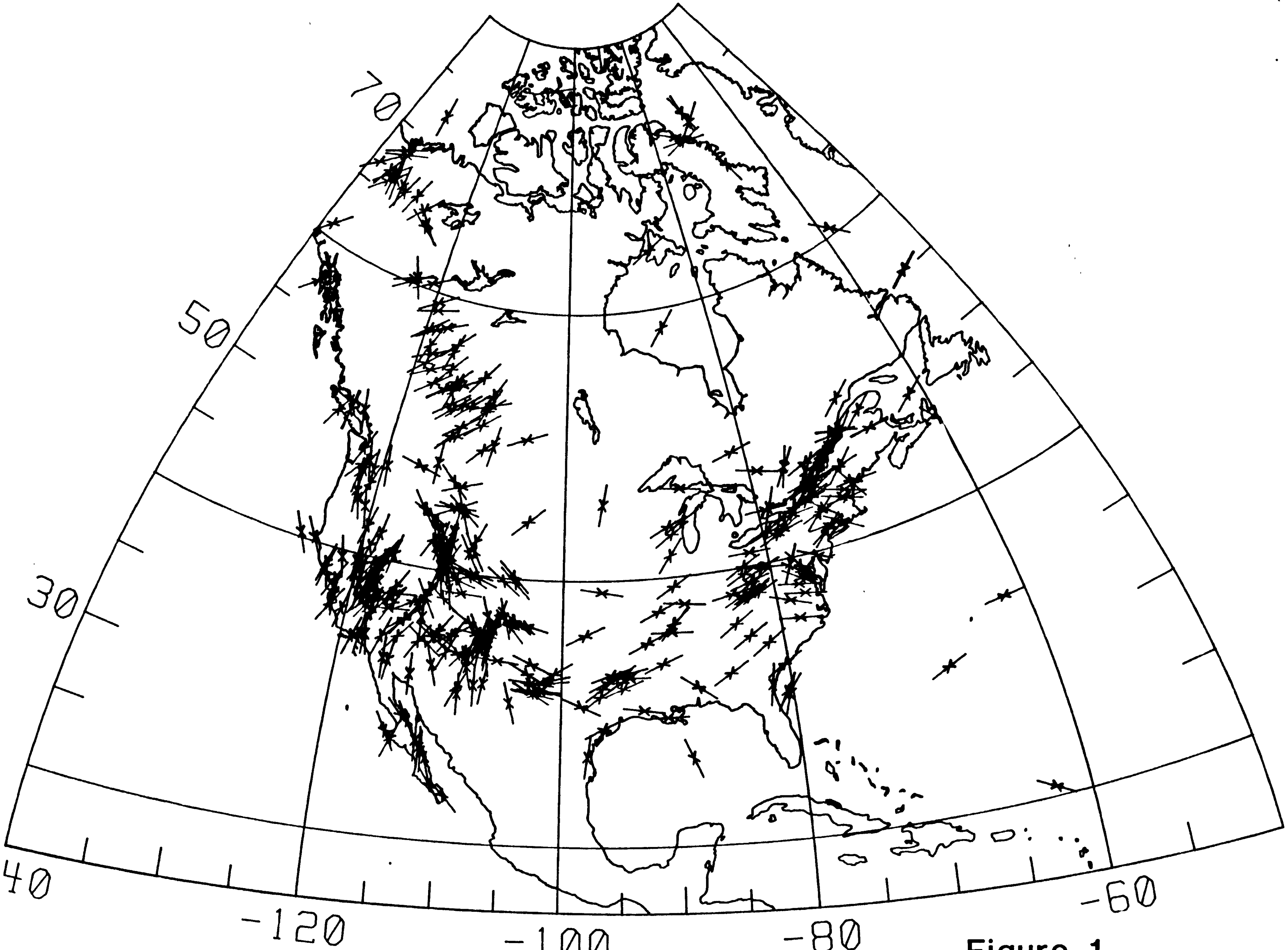

Figure 1

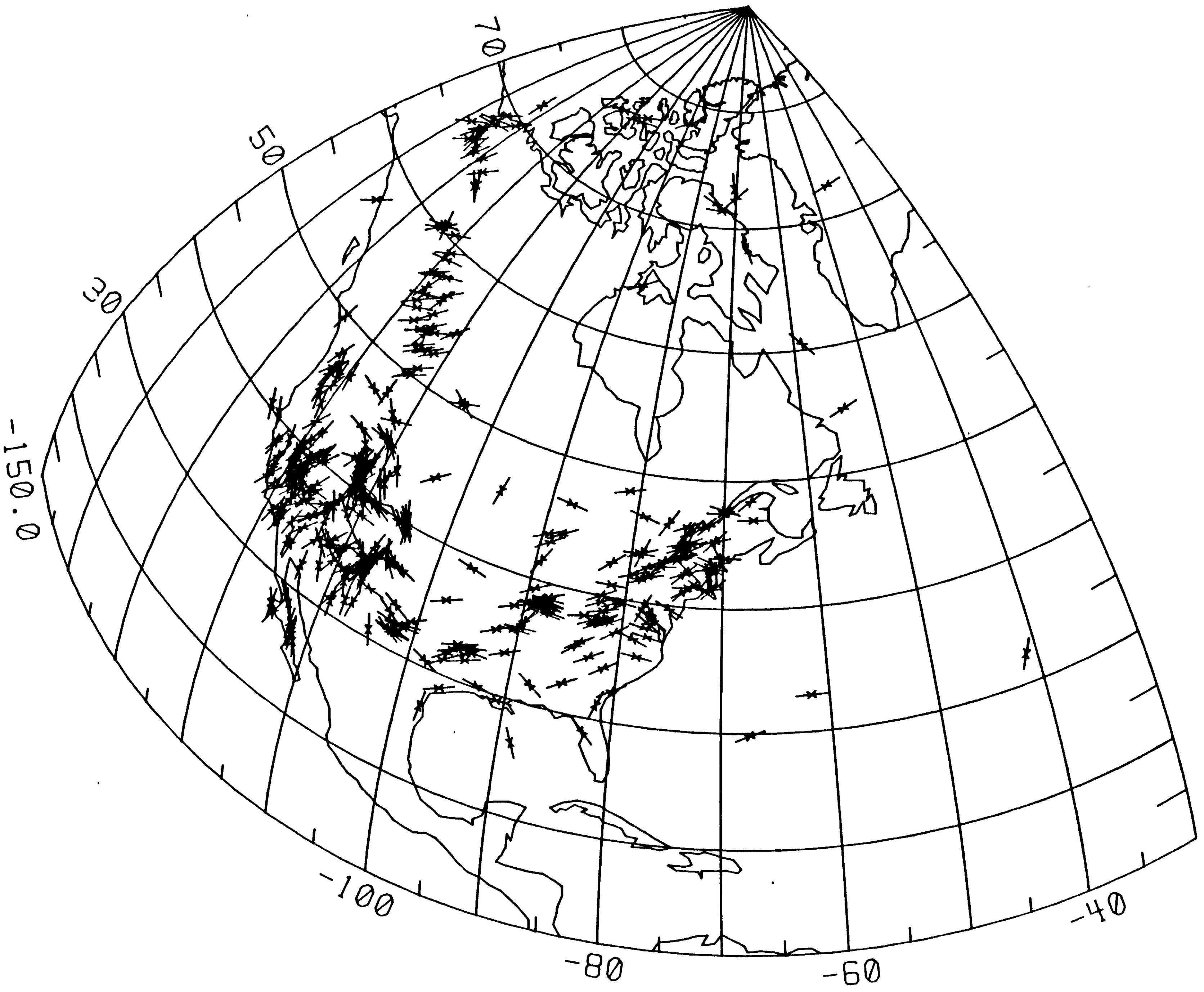

Figure 2

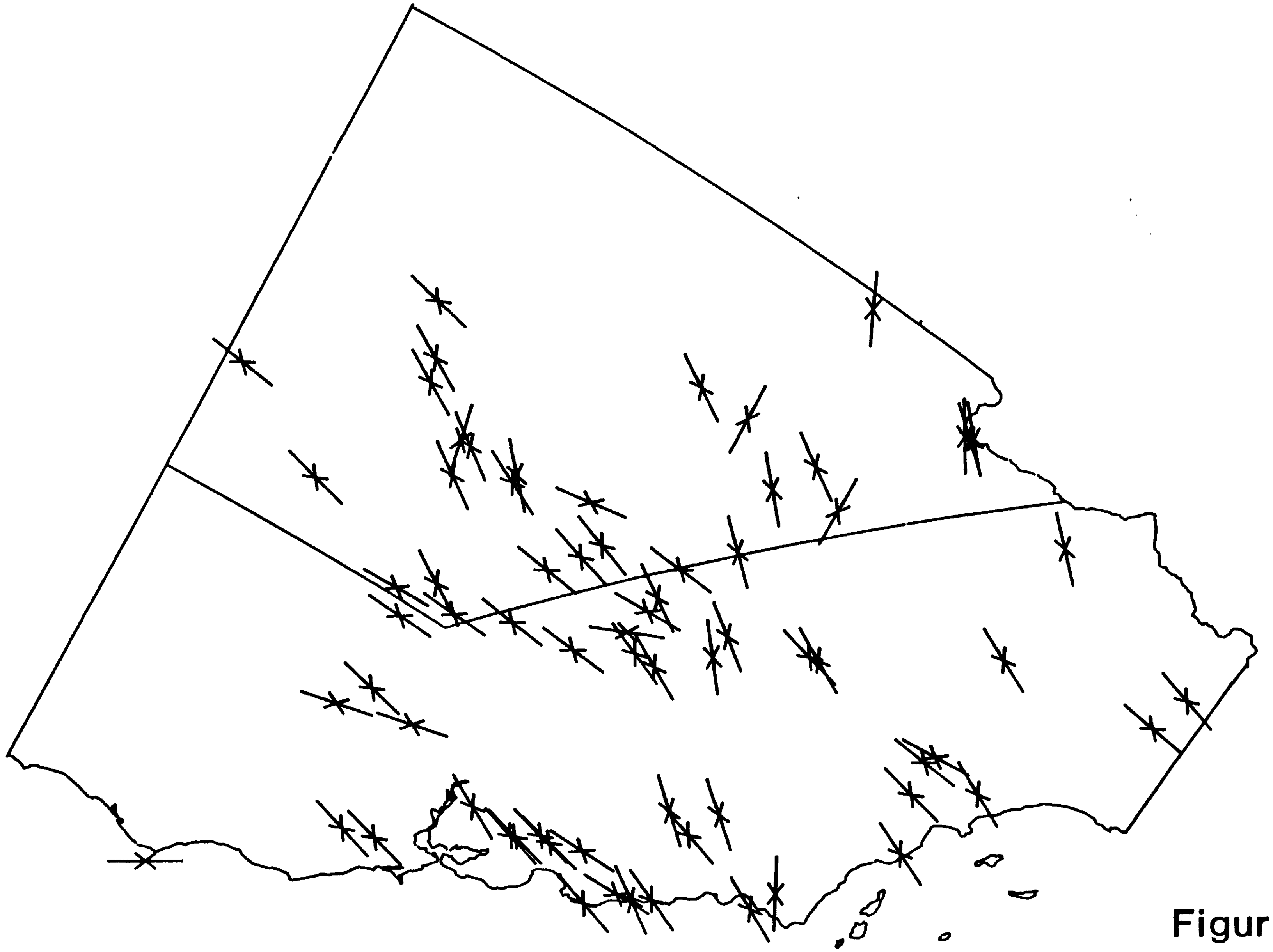

Figure 3

```
AL  1    31.650   -87.910 305 N     G-FS(G)
AK 20    57.012  -135.767  40 SS?   G-VA
AZ  1    31.450  -109.300  28 N     G-VA
AZ  2    32.120  -113.500   0 N     G-VA
AZ  3    33.000  -111.700 343 N     IS-DC
AZ  4    34.000  -109.500 295 N?    G-VA
AZ  5    34.020  -110.600  27 N?    G-VA
AZ  6    34.580  -113.210  32 N     G-FS(G)
AZ  7    35.080  -111.960 305 N?    G-VA
AZ  8    35.250  -111.420 300 N     G-VA
```

Figure 4

```
 -87.910   31.650 305.000
-135.767   57.012  40.000
-109.300   31.450  28.000
-113.500   32.120   0.
-111.700   33.000 343.000
-109.500   34.000 295.000
-110.600   34.020  27.000
-113.210   34.580  32.000
-111.960   35.080 305.000
-111.420   35.250 300.000
```

Figure 5

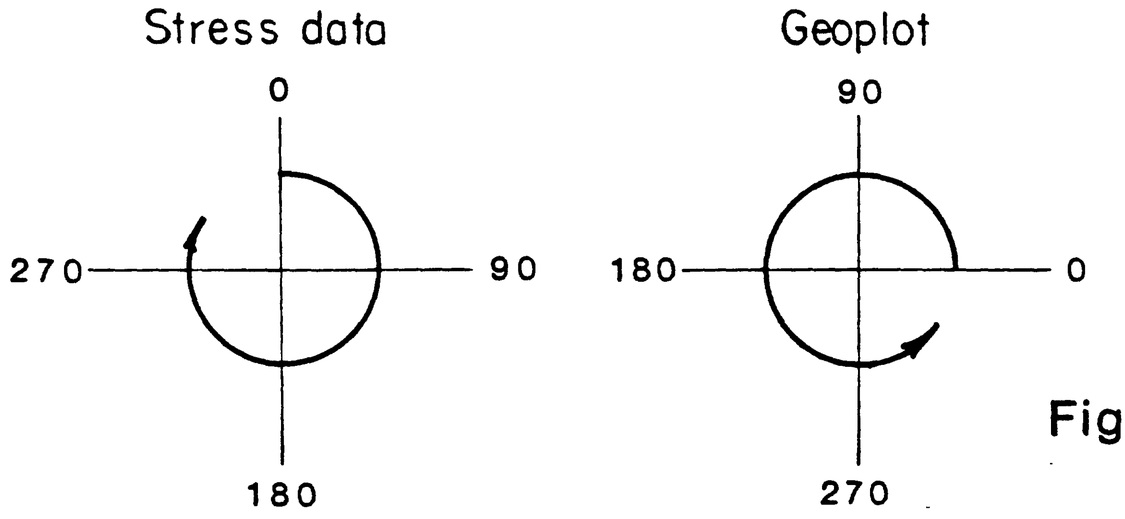

Figure 6