USERS GUIDE

GEOLOGIC SPATIAL ANALYSIS (GSA) PROGRAMS

REMOTE GEOLOGIC ANALYSIS PROJECT

GEOSCIENCES DEPARTMENT
EARTH AND ENVIRONMENTAL SCIENCES CENTER
PACIFIC NORTHWEST LABORATORY

DEVELOPMENTAL SOFTWARE

30 August 1990

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

MASTER

**USERS GUIDE WRITTEN BY
DR. RICHARD THIESSEN
GEOLOGY DEPARTMENT 2812
WASHINGTON STATE UNIVERSITY
PULLMAN, WA 99164-2812**

TABLE OF CONTENTS

1.00 APPLICATION PROGRAMS	7	
1.01 DOCUMENTATION FOR APPLICATION PROGRAMS		8
1.01.01 DEM_INPUT - DEM_input.	8	
1.01.02 VALLEYS - Valleys	9	
1.01.03 VECTORS - Vectors	10	
1.01.04 VECTORS - Tand	11	
1.01.05 VECTORS - Least_squares	12	
1.01.06 COPLANEV - Coplanev	12	
1.01.08 COPLANEV - Process_rv	13	
1.01.09 COPLANEV - Other subroutines	14	
1.01.10 SORT_COPLANE - Sort_coplane	14	
1.01.11 SORT_COPLANE - Sort	15	
1.01.12 SORT_COPLANE - Int_heap_sort3	15	
1.01.13 SORT_COPLANE - Read_cop_stk_dip	16	
1.01.14 SORT_COPLANE - Write_output_files	16	
1.01.15 GET_STKDIP - Get_stkdip	16	
1.01.16 PRINT_STKDIP - Print_stkdip	18	
1.01.17 COPOUT - Copout	19	
1.01.18 COPOUT - cop_window	21	
1.01.19 COPOUT - Cop_ast	21	
1.01.20 COPOUT - Save_copdat	22	
1.01.21 COPOUT - Message	22	
1.01.22 COPOUT - Print_stk_dip	22	
1.01.23 COPOUT - Other_clean_ast	23	
1.01.24 COPOUT - Clean_switch	23	
1.01.25 COPOUT - Clean_do_it	23	
1.01.26 COPOUT - Un_clean_other	24	
1.01.27 COPOUT - Get_min_vec	24	
1.01.28 FORMAT_VIEWTOPO - Format_viewtopo	25	
1.01.29 FORMAT_VIEWTOPO - Read_cop	25	
1.01.30 FORMAT_VIEWTOPO - Read_ps	26	
1.01.31 FORMAT_VIEWTOPO - Write_output	26	
1.01.32 FORMAT_VIEWTOPO - Int_heap_sort	26	
1.01.33 VIEWTOPO - Viewtopo	26	
2.00 UTILITY LIBRARIES	27	

2.01.01	GSA_MODULE - GSA_module	28
2.01.02	GSA_MODULE - Get_input_file_name	28
2.01.03	GSA_MODULE - Get_output_file_name	28
2.01.04	GSA_MODULE - Lower_case1	29
2.01.05	GSA_MODULE - Read_DEM	29
2.01.06	GSA_MODULE - Read_DEM_line	30
2.01.07	GSA_MODULE - Write_DEM_header	30
2.01.08	GSA_MODULE - Write_DEM	31
2.01.09	GSA_MODULE - Printer_file_portion	31
2.01.10	GSA_MODULE - Display_raster	31
2.01.12	GSA_MAPSTART - GSA_mapstart	32
2.01.13	GSA_MAPSTART - Areaset	33
2.01.14	GSA_MAPSTART - Listcodes	33
2.01.15	GSA_MAPSTART - Newcode	33
2.01.16	GSA_MAPSTART - Intitle	34
2.01.17	GSA_MAPSTART - Code_convert	34
2.01.18	GSA_MAPSTART - Start_maps	34
2.01.18	GSA_CREATED:1988-1989	35
2.01.20	GSA_MAPSTART - Newcolor	35
2.01.21	GSA_MAPSTART - Erase_graphics	35
2.01.22	GSA_MAPSTART - Store_graphics	36
2.01.23	GSA_MAPSTART - Read_graphics	36
2.01.24	GSA_MAPSTART - Init_graphics	36
2.01.25	GSA_MAPSTART - Draw_point	37
2.01.26	GSA_MAPSTART - Find_files	37
2.01.27	GSA_MAPSTART - Limit_strs	37
2.01.28	GSA_HEADER - GSA_header	38
2.01.29	GSA_HEADER - Read_GSA_header	39
2.01.30	GSA_HEADER - Write_GSA_header	39
3.00	PROGRAM ANALYSIS SEQUENCE	39
4.00	PERSONAL WHO HAVE WORKED ON GSA	42
5.00	ANCILLARY SOFTWARE PACKAGES	45
5.00.01	GRID_GENERAT - Grid_generate	45

1.00 APPLICATION PROGRAMS

The development of the software discussed in this report has been sponsored and funded by U.S. Department of Energy Office of Arms Control and Office of Basic Energy Sciences, under Department of Energy Contract DE-AC06-76RL0 1830. The programs were rewritten under sponsorship from the Department of Energy facility Morgantown Energy Technology Center in order to develop versions of the code that are more user friendly and utilize more straightforward algorithms.

The main function of the Remote Geologic Analysis (RGA) research project is to generate, test, and implement a series of software packages that utilize digital data bases and automated pattern recognition to detect evidence for control of topography by geologic structures.

The initial test RGA program package was the Geologic Spatial Analysis (GSA) project, which was designed to examine digital elevation models (DEMs) for valley low points that might be controlled by geologic structures (fractures). The programs find alignments of these valleys to define vectors, then check for pairs of vectors that are coplanar in 3D space in order to define the strike and dip as well as location of the potential fault plane.

The GSA programs were initially developed by Dr. Jay Eliason, now at Geologic Analysis and Consulting Services in Deary, ID, between 1980 and 1984. At the time, Dr. Eliason, an employee of Pacific Northwest Laboratory was on educational leave to Washington State University under advisement by Dr. Richard Thiessen. From 1984 to 1987, Dr. Eliason further developed the programs at PNL with assistance from Dennis Beaver and Larry Gerhardstein. In 1987, Dr. Eliason retired, and Dr. Michael Foley took over as project manager for the further development of the RGA program. Dr. Richard Thiessen re-wrote the first several programs in the analysis sequence in order to make the routines more efficient and straightforward, as well as adding internal documentation graphics capabilities, and making them user friendly. Several of the previous programs were combined into single programs.

This report has been produced through automatic extraction of internal documentation blocks included in software source files using the program MAKEDOC, written by Larry Gerhardstein, PNL.

1.01 DOCUMENTATION FOR APPLICATION PROGRAMS

1.01.01 DEM_INPUT - DEM_input

FUNCTION: Inputs DEM files from a disk file and then selects a portion of the DEM to display and to save to a file for analysis with a number of other GSA programs. Also converts U.S. Geological Survey and Defence Mapping Agency DEM files to a format that can be used by the GSA programs. Also allows for merging of adjacent DEM files, repairing of "tears" (data gaps) and spikes, X-Y translation of data, justifying (aligning) an edge, skewing the image, and subsampling DEM.

INPUT: Pre-existing DEM data files on disk in either the old unformatted format or the new formatted ones that include title, size, pixel sizes and other data about the DEM itself (.topo files). Conversely, one can input U.S. Geological Survey 1:24,000 or Defense Mapping Agency 1:250,000 scale DEM files that have been previously copied from the distribution tape to a disk file (MAP.N00----- files).

OUTPUT: DEM files (.topo files) in two possible formats. One is the older version of unformatted DEM files used by most of the PNL programs. The other is the formatted file format used by the GSA program package re-written by R. Thiessen for export to other DOE researchers

LINKAGES: The program has to be linked with the GSA_MODULE and GSA_MAPSTART:
LINK DEM_INPUT, GSA_MODULE,

MAPSTART

AUTHOR: Richard Thiessen, WSU

CREATED: July 1990

HISTORY: Crudely based on the DEMTAP program written
by Harlan Foote and L.Guy Mcwethy,PNL

MODIFIED: No modifications yet

1.01.02 VALLEYS - Valleys

FUNCTION: Detects low points (possible valleys) in digital topographic arrays using a 7 by 7 node detector kernel. The kernel checks if the centerpoint pixel is lower then the end point ones in 16 independent directions. The center must be lower then a threshold smoothing factor input by the user. The number of times the center point is lower then the end points as well as the local trend of the low point can be plotted as printer graphics files or on UIS graphics windows on the computer terminal. The program also gives the option to perform a line narrowing (worm) filtering to the detected valley low points.

INPUT: Digital Elevation Model (DEM) data in either the unformatted version (.TOPO) utilized at PNL and a header file (.TOPO2) if it exists or the new formatted version input with the DEM_INPUT program (.ELEV).

OUTPUT: A .VALLEYS file containing the valley strings for input into the VECTORS program

LINKAGES: The program must be linked with the GSA_MODULE and GSA_MAPSTART program modules

HISTORY: Started as the THALWEG and STRING programs

AUTHOR: Originally written by Jay R. Eliason
then PNL, now at GACS

CREATED: July 1983

MODIFICATIONS: Modified by Dennis E. Beaver then of PNL
between Feb. 1987 and April, 1988
in order to be more compatible
with other GSA programs

RE-WRITTEN: The program was totally re-written by:
Richard L. Thiessen, WSU
July, 1990
in order to create a user friendly and
fully documented version of the code for
export to other Department of Energy
users. The program structure was also
totally rewritten at this time in order
to be more straightforward and efficient.

1.01.03 VECTORS - Vectors

FUNCTION: Program takes valley strings and gives the user
an option to ignore short ones. The strings are
then broken into segments. If adjacent segments
are within an operator specified trend and plunge,
then the segments are joined into a single valley
bottom vector. If not, vectors are made out of
the individual string segments. The vectors trend,
plunge, and location in 3D space can be determined
by just using the end points of the string segment
to define the vector, or the vectors can be
calculated based upon a least-squares routine
using all the valley bottom data points in the
string segment.

INPUT: Valley bottom string data file (.VALLEYS)
which has been created by the VALLEYS program

OUTPUT: A file containing the valley vectors (.VECTORS)
for input into the COPLANEV program

LINKAGES: The program must be linked with the GSA_MODULE
and GSA_MAPSTART program modules

HISTORY: Started as the VECTOR, VECFIT, and

LIMIT_VECTOR_LENGTH programs

AUTHOR: VECTOR originally written by Jay R. Eliason,
formerly at PNL, now at GACS

CREATED: March 15, 1984
VECFIT, LIMIT_VECTOR_LENGTH written by Dennis Beaver,
formerly of PNL

MODIFICATIONS: Modified by Dennis E. Beaver formerly of PNL
between Feb. 1987 and April, 1988
in order to be more compatible
with other GSA programs

RE-WRITTEN: The programs were totally re-written and
combined by Richard L. Thiessen, WSU
July, 1990
in order to create a user friendly and
fully documented version of the code for
export to other Department of Energy
users. The program structure was also
totally rewritten at this time in order
to be more straightforward and efficient.

1.01.04 VECTORS - Tandp

FUNCTION: This routine calculates the length, trend,
and plunge of a vector given the coordinates
of its two end points

AUTHOR: Jay Eliason, formerly of PNL, now GACS

CREATED: 1982-1984

RE-WRITTEN: Richard Thiessen
August, 1990
In order to include more straightforward
routines and internal documentation.

1.01.05 VECTORS - Least_squares

PURPOSE: To fit a vector through string segments using a least-squares linear regression in the 3 planes XY, XZ, and YZ. Do the regression on the variable with the least amount of variation. See Davis, J.C., 'Statistics and Data Analysis in Geology' 2nd ed., chpt. 2 and pp. 176-186 for background.

LAST UPDATE4/15/87

AUTHOR: Dennis E. Beaver

1.01.06 COPLANEV - Coplanev

FUNCTION: Program takes vector data and examines each pair of vectors in three-dimensional space in order to see if all four end points of the two vectors lie within a single plane (within user input strike and dip tolerances). If they do, then the two vectors are coplanar, and may be controlled by the same geologic structure, such as a fault or fracture plane. The plane's strike and dip as well as its location in three-dimensional space is calculated.

INPUT: A file with the vectors in it (.VECTORS extension) created with the VECTORS program.

OUTPUT: A file containing the coplanar detections (.COP) which is used as input into the SORT_COPLANE, GET_STKDIP, and COPOUT programs.

LINKAGES: Program must be linked with GSA_MODULE and GSA_HEADER program modules

COMMON BLOCK:Must be on same directory as COPLANEV.COMMON when it is compiled

HISTORY: Started as COPLANEV program

AUTHOR: Original COPLANEV program written by Jay Eliason formerly of PNL, now at GACS

CREATED: 1982-1984

RE-WRITTEN: Totally re-written by L H Gerhardstein, PNL
29 January 1988

MODIFICATIONS: By Richard Thiessen, WSU
August, 1990
in order to be compatible with other
GSA programs modified at that time

1.01.07 COPLANEV - Process_all

FUNCTION: Allows the coplanar analysis to be done to all
the combinations of vector pairs automatically

1.01.08 COPLANEV - Process_int

FUNCTION: Allows the coplanar analysis to be done
interactively vector by vector. One can
vary the tolerances at any time as well.

1.01.09 COPLANEV - Other subroutines

FUNCTION: A number of other subroutines that
input/output data to files,

1.01.10 SORT_COPLANE - Sort_coplane

FUNCTION: To sort the COPLANE output file (.COP) by strike and dip, creating a sorted index file (.SORT) and a cross index file (.CROSS). Also creates the plane set file (.PS), initializing it with 0's.
This program calculates needed memory, allocates the virtual memory, then calls the sorting routine.

INPUT: A COPLANE file (.COP)

OUTPUT: Files (.SORT), (.CROSS), and (.PS)

LINKAGES: The program must be linked with the GSA_MODULE and GSA_HEADER program modules

WRITTEN BY: Dennis E. Beaver, then at PNL

CREATED: 1/26/88

LAST UPDATE:4/6/88

HISTORY: Started as the SORT_COPLANE program

MODIFICATIONS:Modified by R. Thiessen, WSU
8/20/90 to match documentation
of other GSA programs and to change process for
obtaining file names

1.01.11 SORT_COPLANE - Sort

PURPOSE: To sort the original data by strike & dip.
Uses Heapsort algorithm. First sort the strikes.
Then sort the dips under each strike.

INPUT: size - integer; # records in main data arrays
max - integer; # dimensions in temp (temporary) arrays
stk_ary() - integer; array of strikes

dip_ary() - integer; array of dips
sorted_indices() - integer; indices from raw data
temp_stk_ary() - integer; temporary array of strikes
temp_dip_ary() - integer; temporary array of dips
temp_sort_ary() - integer; temporary array of indices

OUTPUT: sort_ok - logical; flag indicating if sort worked
also passes back the sorted sorted_indices() array

AUTHOR: Dennis E. Beaver, formerly of PNL

LAST UPDATE: 2/29/88

1.01.12 SORT_COPLANE - Int_heap_sort3

FUNCTION: This version is for sorting an integer ra() of length n into ascending numerical order using the Heapsort algorithm, while making the corresponding rearrangement of arrays rb() and rc().

AUTHOR: Subroutine int_heap_sort3 is taken from "Numerical Recipes: The Art of Scientific Computing", 1986, Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T., Cambridge University Press, p.231.

MODIFICATIONS: Modified by Dennis Beaver in order to work with the SORT_COPLANE program

1.01.13 SORT_COPLANE - Read_cop_stk_dip

PURPOSE: To read the .COP file, filling the stk() and dip() arrays.

METHOD:

INPUT:

OUTPUT:

PROGRAMMER:Dennis E. Beaver

CREATED: 1/26/88

LAST UPDATE:2/2/88

1.01.14 SORT_COPLANE - Write_output_files

PURPOSE To create output files .SORT and .PS.

METHOD

INPUT

OUTPUT

PROGRAMMERDennis E. Beaver

CREATED 1/26/88

LAST UPDATE4/6/88

1.01.15 GET_STKDIP - Get_stkdip

FUNCTION: This program reads the .COP file, and the .PS file if necessary, and produces an unformatted strike/dip data file of the coplanar plane detections. One can either work with all coplanar detections output by COPLANEV, or one can examine just those planes assigned to a plane set during the running of COPOUT. The output file is input to the PRINT_STKDIP program, which gives the user data files of arrays of the number of coplanar detections at each planar strike and dip combination (by 1 degree steps). These

latter files are often used to locate concentrations of orientations of coplanar detections. Notice that GET_STKDIP and PRINT_STKDIP are optional programs to the whole GSA analysis. The same clusters of data can be located using the strike and dip grid in the program COPOUT, and/or the sternite plotting option of COPOUT.

INPUTS: Coplanar file (.COP) and plane number (.PS) files

OUTPUTS: An unformatted file (.STKDIP) containing the strikes and dips of

LINKAGES: The program must be linked with the GSA_MODULE and GSA_HEADER program modules

AUTHOR: Dennis E. Beaver, then at PNL

CREATED: 4/6/88

HISTORY: Started as the GET_STKDIP program

MODIFICATIONS: Modified by Richard Thiessen, WSU
7/20/90
during GSA rewrite to get file names and user information blurb.

1.01.16 PRINT_STKDIP - Print_stkdip

FUNCTION: This program reads the .STKDIP unformatted file created by the GET_STKDIP program of the strikes and dips of the colinear detections. The program prints out three files that contain the number of coplanar hits at each strike and dip combination (by 1 degree increments). The three files are for three different dip ranges, with the .LOWDIPS file containing planes that dip from 1 to 30 degrees, .MEDDIPS contains 31 to 60 degrees and .HIGHDIPS contains 61 to 90

degrees. Many GSA analysts use copies of these files to visually locate orientation concentrations of coplanar detections. This program is totally optional to the GSA analysis. The program, COPOUT, which is the interactive program for selecting plane sets of coplanar detections, displays a similar table of number of detections for each strike versus dip combination, and some workers work directly with those tables. Other workers use the sternite plotting option of COPOUT to visually locate orientation concentrations. The GET_STKDIP and PRINT_STKDIP programs can be run before or after COPOUT. If one runs them afterwards, then one has the option of printing out those coplanar detections that have been selected as members of plane sets (assigned to a specific plane set).

INPUT: An unformatted file (.STKDIP) containing the strikes and dips of the coplanar detections.

OUTPUT: Three files containing the number of detections from COPLANEV at every 1 degree strike and dip increment. These files, .LOWDIPS, .MEDDIPS, and .HIGHDIPS can be printed or edited.

LINKAGES: The program must be linked with the GSA_MODULE and GSA_HEADER program modules

HISTORY: Started as the PRINT_STKDIP program

AUTHOR: Originally written by Dennis Beaver, then at PNL

CREATED: 4/6/88

MODIFICATIONS: Modified by Richard Thiessen, WSU
8/20/90 in order to have internal documentation
and change input/output file naming technique

PROJECT MANAGER: Michael Foley (PNL)

1.01.17 COPOUT - Copout

FUNCTION: COPOUT is the main GSA analysis tool for the manual detection of sets of coplanar planar alignments. The basic idea of COPOUT is to allow the user to find groups of coplanar plane sets that have about the same orientation and also align with each other in three dimensions. If one, for instance, finds an orientation in which 12 valley bottoms all plot along the same trend line, then one has probably found a significant plane set, and will want to save it for later analysis. It is usually best to start with plane sets with high ones below about 45 degrees are very prone to erroneous coplanar detections. The program displays a grid of the number of coplanar detections for each strike and dip combination in 1 degree increments, and the user interactively selects orientations. As these are selected, the vectors that make up each plane are plotted on either a map or oblique view (such as right down the plane set orientation). The user can plot the connecting planes in order to see clusters of data. The basic analysis format is to find a cluster of planar orientations, select on each strike and dip combination in the cluster, and look at the map of the resulting planes. If the cluster ends up plotting as tight grouping of planes in the map view, then a single major plane set is being detected. One can click on the orientations again, and they will be removed from the plane set. In this way, the user can manually clean up his plane set to get the most optimum one.

Once a plane set is as optimum as possible, the user may want to obtain a copy of the planes in that plane set, as this is the best opportunity to do so. The user then assigns the vectors to that plane set, and these vectors are cleaned from the data base. Any other coplanar hits that involve these vectors are wiped out. The user then proceeds to the next plane set, and continues until all major orientation clusters have been analyzed.

INPUT: Coplanar detections (.COP), the sorted versions of the detections (.SORT), a cross reference index file of the detections (.CROSS) and a plane selection file (.PS). The latter is empty when the user starts the COPOUT program, and is filled during the program run with the plane set

that each selected coplanar detection is assigned to.

OUTPUT: COPOUT fills in the plane selection (.PS) file. The program also creates screen graphics of plane sets as they are selected in vertical or oblique views, stereonet of the planes, and maps of the coplanar vectors. A file of stereonet data (.STEREO) is created whenever a stereonet is plotted. When one exits COPOUT, one can save the data as a set of files named the same, but with _LAST.COP, _LAST.PS, etc endings

LINKAGES: The program must be linked with the GSA_MODULE and GSA_HEADER program modules

COMMON: Has to be on same directory as its common block, COPOUT.CMN

HISTORY: Started as the COPOUT program

AUTHOR: Dennis E. Beaver, then at PNL

CREATED: Spring, 1988
Last update 5/10/88

MODIFICATIONS: Richard Thiessen, WSU
7/20/90 to print out info to user and also documentation for users guide.

1.01.18 COPOUT - cop_window

FUNCTION: To create a window for output of strike, dip, and count data. The window has mouse capability for selection of specific strike/dip ranges. Uses UIS calls, and does all placement of data and objects in terms of the cell dimensions of the character font used, number of lines of data to be displayed in the window, and number of data cells

[spaces]. At all entry points a check is made to see if the stop_dialog window is active; if it is, the subroutine is exited.

AUTHOR: Dennis E. Beaver, then at PNL

LAST UPDATE:6/30/88

1.01.19 COPOUT - Cop_ast

FUNCTION: Contains ast service routines to handle mouse motion and button selection for main window.

ENTRY POINTS:COP_BUTTON mouse button status change
COP_MOUSE mouse position change
COP_LEAVE mouse moved outside window
COP_TIMER set when button pressed in scroll arrow

INPUT Mouse motion or button pushing.

OUTPUT Mouse position or button state.

AUTHOR Dennis E. Beaver, then of PNL,
with a lot of help from L.H. Gerhardstein, PNL

LAST UPDATE6/30/88

1.01.20 COPOUT - Save_copdat

FUNCTION: To save the data set being worked on.

AUTHOR: Dennis E. Beaver, then of PNL

LAST UPDATE2/8/88

1.01.21 COPOUT - Message

FUNCTION: To alert user to problems
Message number is passed in, appropriate action is taken.

INPUT: mes - integer; message number

OUTPUT: The appropriate message.

AUTHOR: Dennis E. Beaver, then of PNL

LAST UPDATE:6/26/87

1.01.22 COPOUT - Print_stk_dip

FUNCTION: To print current strike/dip data.

INPUT:

OUTPUT: File to the laser printer.

AUTHOR: Dennis E. Beaver

LAST UPDATE:1/29/87

1.01.23 COPOUT - Other_clean_ast

FUNCTION To create a dialog box when other_clean botton is selected. Contains ast service routines to handle mouse motion and button selection in the window.

ENTRY POINTS:CLEAN_BUTTON mouse button status change
CLEAN_MOUSE mouse position change
CLEAN_LEAVE mouse moved outside window

INPUT Mouse motion or button pushing.

OUTPUT Sets logical variable ster_net.

PROGRAMMERDennis E. Beaver

LAST UPDATE4/1/87

1.01.24 COPOUT - Clean_switch

FUNCTION: To act as a switch for other clean operations.

METHOD: Just look and see.

INPUT Ast being fired off.

OUTPUT Appropriate subroutine call.

AUTHOR: Dennis E. Beaver, formerly of PNL

LAST UPDATE5/4/87

1.01.25 COPOUT - Clean_do_it

FUNCTION: To implement the 'other_clean' option. This method involves a user selected vector separation distance which will probably be most effective if it is less than or equal to about 3-4 nodes. The idea is that at this point in the analysis the only planes left are shallow dipping and the best criteria for deciding whether the plane is valid or not is the separation distance between the vectors; ie, if the vectors are close together they are more likely to define a 'real' planar surface such as bedding.

OUTPUT Depends on value of option.
 1 - plot stereonet
 2 - plot vectors
 3 - save the new assignment to ps_ary()

AUTHOR: Dennis E. Beaver, then at PNL

LAST UPDATE6/25/87

1.01.26 COPOUT - Un_clean_other

FUNCTION: To set all ps_ary(i) elements that are 999 or -999 back to 0. Loops through ps_ary().

INPUT size - integer; size of cop_ary() & ps_ary()
 ps_ary - integer; address of ps_ary()

OUTPUT New ps-ary().

AUTHOR Dennis E. Beaver

LAST UPDATE4/7/87

1.01.27 COPOUT - Get_min_vec

FUNCTION: To determine the minimum vector

AUTHOR: Dennis Beaver, then of PNL

1.01.28 FORMAT_VIEWTOPO - Format_viewtopo

FUNCTION: This program reads the cleaned plane data files and creates an unformatted file containing plane set numbers, strike, dip, and coordinates for the locations of the planes in three dimensional space. This file is to be used in the VIEWTOPO program which allows the user to view the cleaned planes in a 3D block diagram. This program also creates a formatted list of planes.

INPUT: A coplanar detections (.COP) and matching plane selections (.PS) files.

OUTPUT: A binary file (.VIEW) in the proper format for the VIEWTOPO program. Also creates an ASCII file, (.PLANES) with a listing of the planes and the plane sets they belong in

LINKAGES: The program must be linked with the GSA_MODULE and GSA_HEADER program modules

HISTORY: Started as the FORMAT_VIEWTOPO program.

AUTHOR: Dennis E. Beaver, then at PNL

CREATED: 1/11/88

LAST UPDATED: 4/6/88

MODIFICATIONS: Richard L. Thiessen, WSU
7/20/90 in order to have information printed to user, change input/output file naming method, and integrate into the GSA user manual documentation.

1.01.29 FORMAT_VIEWTOPO - Read_cop

FUNCTION: Read a .COP coplanar file

1.01.30 FORMAT_VIEWTOPO - Read_ps

FUNCTION: Read a plane selection file

1.01.31 FORMAT_VIEWTOPO - Write_output

FUNCTION: Write the output file.

1.01.32 FORMAT_VIEWTOPO - Int_heap_sort

FUNCTION: Sorting an integer array

AUTHOR: Subroutine int_heap_sort is taken from
"Numerical Recipes: The Art of
Scientific Computing", 1986, Press, W.H.,
Flannery, B.P, Teukolsky, S.A., and
Vetterling, W.T., Cambridge University Press, p.231.

1.01.33 VIEWTOPO - Viewtopo

FUNCTION: Views the topography and selected planes in three-dimensions from any look angle. Has a number of options, including plotting the topography as a net, showing the plane interception with a bounding cube, showing the plane interception with the topographic surface of the earth, and showing multiple planes, including their intersection. Can also plot two windows simultaneously which create a stereopair.

INPUT: A file with the topography (.TOPO) and a file with the selected planes (.VIEW).

OUTPUT: No files are output, but the program does create three-dimensional displays of the topography and planes.

LINKAGES: The program must be linked with the GSA_MODULE and GSA_MAPSTART program modules

AUTHOR: L H Gerhardstein, PNL

CREATED: 3/2/88

LAST UPDATED: 3/5/88

MODIFICATIONS: Richard Thiessen, WSU
7/20/90 in order to work with GSA program user guide,

and some internal documentation as well.

SUBROUTINES: There are a number of subroutines with this program that aid in the input and display of the graphics data.

2.00 UTILITY LIBRARIES

Chapter 2 contains documentation of the utility software libraries which are part of the geologic spatial analysis (GSA) software package. These libraries need to be linked to various of the GSA programs during compilation, as stated in the documentation for each program under LINKS:

2.01 DOCUMENTATION FOR UTILITY LIBRARIES

2.01.01 GSA_MODULE - GSA_module

PURPOSE: This is a series of subroutines that are used by the Geologic Spatial Analysis (GSA) software package. This module needs to be linked with all the GSA programs if they are recompiled.

LINKS: In order to get an object file of GSA_MODULE, you need to have previously typed:
FORTRAN GSA_MODULE

AUTHOR: Richard L. Thiessen, WSU
(unless specified otherwise)

CREATED: July and August, 1990

2.01.02 GSA_MODULE - Get_input_file_name

PURPOSE: Based on a model generated with a file extension, the program gives a directory of files that are on the specified disk. The user then types in the file name. Files can be on external disk drives or tapes, as long as they are mounted and available to the operator's terminal.

AUTHOR: Charles Cole, PNL

MODIFICATIONS: Greatly modified by R. Thiessen, WSU
7/9/90 to incorporate into GSA software programs.

2.01.03 GSA_MODULE - Get_output_file_name

PURPOSE: Subroutine takes the input file name and separate extension and determines if user wants to use them to automatically name the output file, or to input a new name.

AUTHOR: Richard L. Thiessen

CREATED: 7/9/90 modelled after one
in the old version of the THALWEG program

2.01.04 GSA_MODULE - Lower_case1

PURPOSE: Subroutine takes single character answers to questions and converts them all to lower case characters

AUTHOR: Richard L. Thiessen

CREATED: 7/6/90

2.01.05 GSA_MODULE - Read_DEM

PURPOSE: This routine allows the user to input the portion of the DEM file that is desired, and then reads in the Digital Elevation Model data. Three different types of files may be used. The most commonly used file is a .TOPO file, which is a binary unformatted file of just the topography data. In July, 1990, Richard Thiessen established a companion header file, designated with a .TOPO2 extension, which is an ascii file containing all the information about the DEM,

including title, source, grid size, dem quality, pixel size and units, coordinates of lower left corner, and minimum and maximum elevations. This file can be edited or printed. If a .TOPO2 header file does not exist for the requested .TOPO file, then this subroutine will request the necessary data, and give the user an option to create the header file. An .ELEV file is a formatted (ascii) file which includes information contained in the .TOPO2 header, followed by all the topography data. This type of file is simply for transporting of data bases to other systems or users, and is not used by the GSA programs in general.

AUTHOR: Richard L. Thiessen

CREATED: Spring of 1989
for his COMPARE program.

RE-WRITTEN: July, 1990 by R. Thiessen
for inclusion in the rewriting of the GSA programs

2.01.06 GSA_MODULE - Read_DEM_line

PURPOSE: this routine reads in a single line of DEM data from a .topo file. This routine is called from READ_DEM so that the whole line is input, but only a part is fed back to the main calling program, i.e, variable arrays can be used to input the central part of a line of data that is wider then the storage array, topo. In other words, dem_imax, set up at beginning of program run, is less than i_end, width of input DEM data base.

AUTHOR: Richard L. Thiessen

CREATED: August, 1990
for inclusion in the rewriting of the GSA programs

2.01.07 GSA_MODULE - Write_DEM_header

PURPOSE: This routine allows the user to output a DEM header file in the new file format (formatted) which includes information such as the size of the grid, pixel spacing and units, and map coordinates of the lower left corner. The file will be a separate file with a .TOPO2 extension, or the first part of an .ELEV (ascii) file

AUTHOR: Richard L. Thiessen

CREATED: July 1990
for inclusion in the rewriting of the GSA programs

2.01.08 GSA_MODULE - Write_DEM

PURPOSE: This routine allows the user to output a DEM file in the new file format (formatted) which includes information such as the size of the grid, pixel spacing and units, and map coordinates of the lower left corner.

AUTHOR: Richard L. Thiessen

CREATED: July 1990
for inclusion in the rewriting of the GSA programs

2.01.09 GSA_MODULE - Printer_file_portion

PURPOSE: Allows the user to print out a printer file of single character symbols (valley trends, number of hits) on an 80 character wide page

AUTHOR: Richard L. Thiessen

CREATED: 7/11/90

2.01.10 GSA_MODULE - Display_raster

PURPOSE: Takes input topography or equivalent data and creates a color raster type image of the data, with the elevation value at each pixel corresponding to a color in a rainbow (blue=low, red=highs)

AUTHOR: Dennis Beaver, formerly at PNL

MODIFICATIONS: Modified and internally documented by Richard Thiessen in 1990.

PURPOSE: So that the size of the displayed raster image output by DISPLAY_RASTER could be adjusted (shrunk, blown up, plot just part of) using variable arrays

AUTHOR: Richard L. Thiessen

CREATED: Aug. 14, 1990

2.01.12 GSA_MAPSTART - GSA_mapstart

PURPOSE: This is a series of subroutines designed to allow for the initiation of map style graphics using UIS graphics.

AUTHOR: Richard L. Thiessen
Geology Department
Washington State +.
Pullman, WA 99164-2812
(unless otherwise designated)

CREATED: 1988-1898

MODIFICATIONS: By R.Thiessen, July and August, 1990
in order be utilized by GSA program package

LINKAGES: This module is linked to other programs with a LINK statement in order to get an object file of GSA_MAPSTART, you need to have previously typed: FORTRAN GSA_MAPSTART

2.01.13 GSA_MAPSTART - Areaset

PURPOSE: This subroutine inputs the part of the map to plot. The user has choices between using the whole study area, the area used for the previous plot, or change the plot area.

AUTHOR: Richard Thiessen

CREATED: 1988-1989

2.01.14 GSA_MAPSTART - Listcodes

PURPOSE: This routine lists out the codes that are currently being used on a plot, calculation, or data output to storage. The codes are used to distinguish different data types, such as lineaments picked by different interpreters.

AUTHOR: Richard Thiessen

CREATED: 1988-1989

2.01.15 GSA_MAPSTART - Newcode

PURPOSE: This subroutine checks to see if an input code (from storage) is a new one, and if so, inputs a title and color to be used with the code

AUTHOR: Richard Thiessen

CREATED: 1988-1989

2.01.16 GSA_MAPSTART - Intitle

PURPOSE: Input a title for the current plot or listing. If only one code is being used, then the user is given the option of using the title associated with that code.

AUTHOR: Richard Thiessen

CREATED: 1988-1989

2.01.17 GSA_MAPSTART - Code_convert

PURPOSE: The actual codes are not stored for each data point. Instead, the number is stored in an array, and the slot in that array that matches the code of the data point is kept with that datum

AUTHOR: Richard Thiessen

CREATED: 1988-1989

2.01.18 GSA_MAPSTART - Start_maps

PURPOSE: This routine initiates the graphics windows, inputs a title for the window, draws the boundary, labels the window, and adds a scale bar.

AUTHOR: Richard Thiessen

CREATED:1988-1989

2.01.19 GSA_MAPSTART - Start_region

PURPOSE: This subroutine does the same as Start_map, but it works for a the entire map region, rather than being able to map part of the region of interest. The operator is not given the option of inputting the area to plot.

AUTHOR: Richard Thiessen

CREATED: 1988-1989

2.01.20 GSA_MAPSTART - Newcolor

PURPOSE: Choose a new pen color for the plots (on screen or plotter)

AUTHOR: Richard Thiessen

CREATED: 1988-1989

2.01.21 GSA_MAPSTART - Erase_graphics

PURPOSE: This subroutine allows the operator to choose which of the existing graphics windows he wants to erase.

AUTHOR: Richard Thiessen

CREATED: 1988-1989

2.01.22 GSA_MAPSTART - Store_graphics

PURPOSE: This routine allows the operator to store a graphics window for later rendering to a plotter or printer file. During the summer of 1990, R. Thiessen disabled the graphics window display list so that the windows can be erased in a short time. As a result, the windows can no longer be stored.

AUTHOR: Richard Thiessen

CREATED: 1988-1989

2.01.23 GSA_MAPSTART - Read_graphics

PURPOSE: This routine reads in a graphics image data file that has been previously stored in order to plot more features on it. Note that as of summer, 1990, R. Thiessen disabled the graphics window display list for fast window erasing, and so the windows cannot be saved and later read back in.

AUTHOR: Richard Thiessen

CREATED: 1988-1989

2.01.24 GSA_MAPSTART - Init_graphics

PURPOSE: This routine sets up the color mapping scheme which will be used to display data. This includes individual colors, black, white, and greys as well as a rainbow of colors for plotting gradational data such as elevations. It also inputs a title for the window, sets up the writing modes for text, enables/disables the graphics window display list, inputs the physical size of the window on the computer monitor, creates the graphics window, checks to see if the operator wants a white versus black background, whether to increase the width of any lines being drawn, and whether to clip the area being drawn.

AUTHOR: Richard Thiessen

CREATED: 1988

2.01.25 GSA_MAPSTART - Draw_point

PURPOSE: This routine draws a line segment to a point

AUTHOR: Richard Thiessen

CREATED: 1988-1989

2.01.26 GSA_MAPSTART - Find_files

PURPOSE: It is designed to find files
that are on a specified directory.

AUTHOR: Charles Cole, PNL

2.01.27 GSA_MAPSTART - Limit_strs

PURPOSE: This routine allows the operator to input limits
to the structures that are going to be plotted,
printed, or stored. For instance, if one has 532
lineaments, one can choose to work
with only those from number 316 to 380.

2.01.28 GSA_HEADER - GSA_header

FUNCTION: This set of subroutines are called to read and
write the header portion of files which follow
the header standard implemented for Remote
Geologic Analysis/Geologic Spational Analysis
(RGA/GSA) programs.

The basic design of the header scheme is:

1. A header record (line) is 32 bytes long.
2. The first 4 bytes of a record is a 4 character id,
the following 28 bytes contain the related data;
data shorter than 28 bytes is padded with 0's.
3. A header can contain an unlimited number of records.
4. Except for the first and last header record, the order
of records is not constrained.
5. The first header record will be: CR16,0,0,0,0,0,0,0
6. The last header record will be: HEND,0,0,0,0,0,0,0

INPUT: Header portion of data files

OUTPUT: Header portion of data files

LINKAGES: This module must be linked to many of the GSA programs, particularly all the ones in the latter part of the analysis sequence. An object file of this module must be obtained by typing
FORTRAN GSA_HEADER

COMMON: Needs the GSA_HEADER.COMMON common block

AUTHOR: Dennis E. Beaver, then at PNL

CREATED: 2/1/88

LAST UPDATED:4/12/85

2.01.29 GSA_HEADER - Read_GSA_header

FUNCTION: This subroutine is a dispatcher, calling the appropriate subroutines for specific file types.

AUTHOR: Dennis Beaver

CREATED 2/1/88

2.01.30 GSA_HEADER - Write_GSA_header

FUNCTION: This subroutine is a dispatcher, calling the appropriate subroutines for specific file types.

AUTHOR: Dennis Beaver

CREATED 2/1/88

3.00 PROGRAM ANALYSIS SEQUENCE

The following is a brief listing of the order in which the GSA programs should be run, and what to do in each one.

1. INPUT_DEM

If you have acquired your topography data bases from the U.S.G.S., then they will have to be converted to the format used by the GSA programs. If they have already been done, you can use INPUT_DEM to mosaic together and/or manipulate the DEM data, such as repair gaps and spikes, move and skew the data, subsample it, and mosaic data. This program produces a color image (with changeable colors) of the DEM data base. Store the resulting data into two files, one with a .TOPO extension that is a binary file with the elevation data, and the other is a .TOPO2 header file with all the required information such as file and pixel sizes, corner coordinates, etc.

2. VALLEYS

This program takes the topography file and finds low points which correspond to valley bottoms. These detections are narrowed using a line narrowing worm filter, and connected into valley strings. The user can use color plots to examine the number of times each valley bottom is detected, as well as trends of the valley bottoms. The individual grid locations are connected to form a valley string. This program replaces the old THALWEG and STRING programs.

3. VECTORS

This program takes the valley strings and segments them into short vectors. Those adjacent segments that are subparallel within user specified trend and plunge limits can be reconnected into longer vectors. The vector locations and orientations are recalculated using a least squares technique. Graphics options allow the various vector combinations to be This program replaces VECTOR, VECFIT, and LIMIT_VECTOR_LENGTH.

4. COPLANEV

COPLANEV takes every pair of vectors and makes two planes. If these planes are nearly coplanar, i.e. within user specified angles (strike and dip) of each other, then the original vectors were also coplanar, and may be defining some planar geologic structure.

5. SORT_COPLANE

This program takes the coplanar output from COPLANEV and sorts it into a format that can be used by the COPOUT program.

6. GET_STKDIP and PRINT_STKDIP (OPTIONAL)

These two optional programs allow you to produce a printer file with the number of coplanar detections at each one degree increment of strike and dip. Orientation clusters of the planes can be identified. A similar plot is generated during runs of COPOUT, and COPOUT also has an option to produce stereonet plots of this information.

7. COPOUT

This is the main interactive analysis tool for the identification of sets of planes that align in location as well as orientation. The user is given a grid of number of coplanar detections that fall within each one degree (by strike and dip) orientation range. The operator uses the mouse to select which orientations he wants, and they are plotted on a map. Planes can be removed from the map, in order to more closely define a single cluster of planes. A stereonet of orientations can be plotted to aid in the analysis.

8. FORMAT_VIEWTOPO

This program simply takes the output from COPOUT and reformats it into a mode compatible with the next program,

9. VIEWTOPO

This interactive program allows the user to plot a three dimensional view of the topographic surface of the earth. Planes that have been defined by GSA can be individually selected for plotting in a range of colors. Their trace on the surface of the earth can be depicted. The program has options to modify perspectives, look direction, vertical exaggeration, etc. A stereopair can be directly drawn on the workstation screen.

4.00 PERSONAL WHO HAVE WORKED ON GSA

The Geologic Spatial Analysis program package has been generated at the following institutions:

PNL	Pacific Northwest Laboratory P.O. Box 999 Battelle Blvd. Richland, WA 99352.
WSU	Geology Department Washington State University Pullman, WA 99164-2812
GACS	Geologic Analysis and Consulting Services P.O. Box 314 Deary, ID 83823

The following personnel have worked on the GSA programs

Dr. Jay Eliason, originator of GSA programs and project. Was formerly at PNL and initiated GSA while on educational assignment to WSU. The original GSA program set was written by Dr. Eliason as his PhD. thesis. He further developed GSA as the project manager at PNL from 1984 to 1987. He is presently at GACS, where he is producing sophisticated GSA analysis systems and computer codes.

Staff at PNL:

Dr. Jay Eliason, former project manager at PNL,
Now at GACS, where he is project manager.

Dr. Michael Foley, present project manager at PNL.

Dr. Richard Thiessen, PNL Affiliated Staff Scientist

Dennis Beaver, now at NORD Resources

suite D

87112

12800 Lomas NE,
Albuquerque, NM,

Larry Gerhardstein

L. Guy Mcwethy

Harlan Foote

Charlie Cole

Pat Heasler

Staff at WSU:

Dr. Richard Thiessen, project manager

Dr. Jay Eliason, doctoral student (1982-1984)

Visiting assistant professor (1985-)

Dr. Michael Foley, Visiting assistant professor (1987-)

Eric Rieken

Douglas Neves

Lisa Johnson (formerly at WSU)

Staff at GACS:

Dr. Jay Eliason, project manager

Valerie Eliason

5.00 ANCILLARY SOFTWARE PACKAGES

Chapter 5 contains documentation of ancillary software packages which are not directly part of the geologic spatial analysis (GSA) software package, but are utilized to enhance and test a GSA analysis.

5.00.01 GRID_GENERAT - Grid_generate

FUNCTION: Program inputs data from a geochemical sampling sites (.SITE) file, a NURE (National Uranium Resource Evaluation) geochemical data file (.NURESITE), or a data file with depths to lithologic layers, as observed in well hole intersections (.CAL) file. The user can then plot the various geochemical or well sites with color coding related to the concentration or depth. The color on the plots can be controlled by a linear color scale, or a logarithmic one to emphasize lower productions/concentrations. The user can examine these plots in order to detect possible fault offsets or alignments of geochemical/production highs. The data can be interactively edited. The program then calculates a gridded surface using user parameters based on one of the data (such as gold content or depth to a lithologic layer). This gridded file can then be displayed and saved.

INPUT: Geochemical sampling sites file (.SITE) and/or NURE geochemical data base (.NURESITE) and/or well hole depth and production data, (.CAL).

OUTPUT: A new sampling sites file (.SITE) and/or a contour grid of one lithologic layer/chemical in a format that can be input into the DEM_INPUT program (.TOPO and .TOPO2 files) plus a text file with the gridding process information, .GRID.

LINKAGES: The program has to be linked with the GSA_MODULE and GSA_MAPSTART:
LINK DEM_INPUT, GSA_MODULE, MAPSTART

RIGIN: Program started out as three separate programs,
SITES, written by R. Thiessen, DEM_INPUT also
written by Richard Thiessen, and the TGRID_GEN
program written by L.Guy Mcwethy, PNL

AUTHOR: Richard Thiessen, WSU

CREATED: August 19-29,1990

PROJECT MANAGER:Michael Foley, PNL

END

**DATE
FILMED**

12/30/91