Pergamon

Computers & Geosciences Vol. 20, No. 1, pp. 23–30, 1994
Copyright © 1994 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0098-3004/94 $6.00 + 0.00

THE REPRESENTATION OF DIRECTIONAL GEOLOGICAL DATA: TEC-HARDWARE INDEPENDENT HIGH-QUALITY GRAPHICS

CARL RICHTER[1]* and DIETER KREJCI[2]
[1]Department of Geological Sciences, University of Michigan, Ann Arbor, MI 48109, U.S.A. and [2]Institut für Geologie, Universität Tübingen, Sigwartstrasse 10, D-7400 Tübingen, Germany

(Received 1 December 1992; revised 8 August 1993)

Abstract—We present an almost hardware independent program collection (language C) for IBM and compatible computers for a high-quality printout of three-dimensional (3-D) geological data. Linear data and poles to planes are plotted in equal area and equal angular stereographic projections as polar points, great circles, or symmetric and asymmetric rose diagrams. Fault–slip data are presented in plots containing (1) the great circle of the fault and the slip direction of the hanging wall or (2) the pole to the fault and the slip direction of the hanging wall. The symbol size and type (fifteen different symbols), legend, head and bottom lines, and projection type can be modified using definition files. Utilities include a program to rotate polar data interactively on the computer screen and a separation program for fault–slip data. The programs support *c.* 50 different graphic cards, and *c.* 40 printers/plotters, and can be used essentially with every hardware configuration. The plots can be converted into HPGL, PIC, GEM, TIFF, or postscript graphic files and thus are compatible with commercial drafting programs.

Key Words: Graphical representation, Orientations, Directional data, Stereographic projection.

INTRODUCTION

Numerous programs and algorithms for the graphical representation of directional data (e.g. bedding planes, fold axes, fault–slip data, anisotropy directions) in stereograms have been developed in the past few years for personal computers (e.g. Griffis, Gustafson, and Adams, 1985; Guth, 1987; Adam, 1989; Charlesworth and others, 1989; Pecher, 1989; Pilant, 1989; van Everdingen, van Gool, and Vissers, 1992). The algorithms to calculate the plot coordinates of equal area (Schmidt net) or equal angular (Wulff net) projections are easy to obtain (e.g. Wallbrecher, 1986), a plot of great-circles and fault–slip data, however, is more involved (e.g. Krejci and Richter, 1991). Many earth scientists use different more or less sophisticated programs for their purposes. A plot on the computer screen can be obtained readily, but an acceptable and publishable output needs substantial programming skills. This is the weak point of many programs. Hardcopies of the graphic screen or plotter output are frequently used. Some recent programs use a laser printer and many solutions offer different plotting symbols, symbol-sizes, great circle representations, or rose diagrams. However, a satisfactory output that can be adopted to individual taste and requirements (e.g. use with a microscope or other lab equipment) usually is difficult to obtain. The hardware (graphics card and printer) is the second problem that may not be treated satisfactorily. Many programs are written for one specific graphics adapter and output device and can be used only with that specific hardware configuration. This makes system upgrades or the work with the same data in different computing environments cumbersome.

In this contribution we offer an almost graphic card independent program collection for IBM compatible computers. 50 different graphic adapters are supported (a selection is shown in Table 1), that can be installed for *c.* 40 different printers/plotters (Table 1), and that can convert data files into graphic files that are compatible with Lotus (*.PIC), Ventura (*.GEM), Hewlett Packard (HPGL), TIF bitmap (TIFF), and encapsulated Postscript files. These files can be imported into commercial drafting programs, which offers the option to modify the graph, combine stereographic projections for example with digitized maps, and obtain a high-quality output of graphs. Our collection of programs contains (1) stereographic projections (equal area and equal angular) of lines or poles to planes (2) rose diagrams (symmetrical and asymmetrical) (3) great circle plots of planes (equal area and equal angular) (4) representation of fault–slip data (pol/slip, Hoeppener, 1955 and great-circle/slip, e.g. Angelier, 1984) (5) a split utility for fault–slip data (Krejci and Richter, 1991) and (6) a rotation utility that rotates data using the arrow keys

*Present address: Ocean Drilling Program, Texas A & M University Research Park, 1000 Discovery Drive, College Station, TX 77845, U.S.A.

on the computer screen. All programs can be used independently or menu driven from an overlay program (TEC.EXE) that includes editors for the input of data and the manipulation of definition files.

The programs are written in C (Turbo-C Version 2.0 or Borland C++ Version 2.0 and 3.0). Hardware independence is obtained by using the graphic toolbox GraphiC (version 6.0; Scientific Endeavors) that handles the hardware installation and the output procedures. It is our intention to provide the routines for a high-quality graphic print of directional data that can be modified for individual taste and needs, such as certain input file formats, sizes, fonts, line thickness, or headlines. Additional plotting routines such as modified stereographic projections, *xy*-plots, or triangle diagrams can be added easily to the program collection and printed using the same output utilities.

INPUT FILES AND PROGRAM ALGORITHM

We will describe the use of TEC, the input file formats, and discuss the application of GraphiC to obtain hardware independence using the source code of our great circle program. Sic programs (Pol.C: stereographic projections; Britt.C: fault–slip data representation; Rose.C: rose diagrams; GCir.C: great-circles; Rotate.C: interactive rotation with arrowkeys on the screen; Rot.C: rotation with input of the rotation axis and angle; Split.C: splitting of fault–slip data) can be used separately under command-line or with a menu program (Tec.EXE), that executes the different plotting programs and contains an editor for the input of data and the management of definition files. The hardware independence is obtained by an installation program (Equip.EXE) that is part of the GraphiC toolbox. Equip defines the output device, the graphic adapter and the location of the subdirectory that contains the programs.

Input text files (ASCII), with the exception of the fault–slip data sets, are identical for all subprograms. The first row in the file indicates whether data are lines or planes ("Li" or "Pl"), the following rows contain dip direction and dip angle of planes, or direction and plunge of axial data, separated by a blank. The file optionally may contain a third column with a number between 0 and 15 that defines the plot symbol. If no plot symbol is defined, the default value (0: square) or the value from a definition file is used. Input files for the representation of fault–slip data contain the dip direction and the angle of dip of the fault planes and the corresponding direction and plunge of the striations. A fifth column contains a symbol for the relative sense of movement of the hanging wall (+ for up, − for down, 0 if ambiguous or not determined). A sixth column may contain optionally a number between 0 and 15 to represent the plot symbol. The procedure "void readinput" can be changed, if other conventions for the definition of the directional data or file formats are preferred. In addition to the data file a definition file (ASCII) can be edited in which the figure captions, head- and bottom lines, symbol sizes and -types, and projection type are defined. If no definition file is available, the program uses the default values. Definition files differ for the subprograms, for example for stereographic projections of poles (Pol.C):

Y	(Plot periphery; N prints only the data and the stereogram)
AREA	(equal AREA projection, ANGULAR is equal angular projection)
Headline	(any text to identify the plot)
Bottomline	(additional information about the graph)
0.1	(symbol size in inches)
2	(symbol style 0–15; e.g. 0: square; 1: circle; 3: cross)

The specific format and options for the definition files can be obtained from (1) the source code, (2) a short information, that appears on the screen if any of the programs is executed without a data file, or (3)

Table 1. Selected list of supported hardware; total list contains *c.* 50 different graphic cards and more than 40 different printers and plotters

Graphics adapter	Printer
EGA mono/color 640 × 480	Epson FX, RX
Genoa Super EGA 800 × 600	Okidata 92, 93
VEGA 640 × 480	IBM Graphic
VGA 640 × 480	Tektronix 4696 inkjet
Tseng ET4000 cards	HP-GL and HG-GL/2 plotters
VESA cards 800 × 600	Epson LQ1500
AT & T 640 × 400	HP Laserjet +
Hercules monochrome	NEC Pinwriter P series
CGA 640 × 200	CANON LBP-811 Laser
Sigma color 400 640 × 400	HP Thinkjet, Paintjet color
Tecmar GM 720 × 696	Epson HI-80 plotter
Toshiba 3100 640 × 400 mono	Seikosha SK-300AI
Texas Instruments TIGA	Toshiba P321 and P351
Trident 8800 highres	Houston DMP-xx ploters
ATI VGA Wonder 800 × 600	Watanable Digi-Plot
Sigma LaserView Plus	Calcomp ColorMaster
Tecmar GM 720 × 696	Postscript (TM) printer

from the built-in definition file editor of the menu-overlay program TEC.

The Appendix gives the complete listing of GCirc.C. The names of the subroutines are self-explanatory and the purpose of the procedure is easy to understand even without being familiar with C. Data are read into memory (void readinput), the optional definition file is read (void readdef), and the graph is plotted (double plot_gcircle) in an equal area net (Lambert) or an equal angular net (Wulff). The entire program listing is extremely short and the actual plotting code is only around 50 lines long. The program plots an existing data file (GCir ⟨filename⟩) and has no menu or any option to modify the graph. This has the following advantages: (1) the program is uncomplicated and fast for routine use; (2) it can be incorporated into the menu driven structure of TEC; or (3) into any other menu-driven program, that uses different subprograms; and (4) the programs can be executed using a batch file to plot a great number of data files without the presence of the user. Modifications from the default output are defined in a separate ASCII file.

The GrapiC procedure "endplot" activates an output menu if the ESC key is hit after the plot is completed. The output menu gives the choice between draft or high-quality prints on the installed output device or the conversion to another graphic file format.

The structure of the other graphic programs is the same and there is no need to describe them in detail here. They differ in the procedure that reads the definition file and in the procedure "plot_gcircle", that is replaced, for example by "plot_poles", "plot_rose", "plot_angle", or "plot_hoeppener". Instead of this procedure any other procedure can be inserted to obtain plots for various purposes.

UTILITIES

The rotation utilities use the rotation matrix (e.g. Wallbrecher, 1987; Vissers and Bollegraaf, 1989) to rotate directional data about a user defined axis on the sphere. We have included two versions: (1) uses the input of a known rotation axes and angle to rotate the data and write them into a new file that can be manipulated further within TEC; (2) rotates data on the computer screen by modifying the rotation matrix interactively with the arrow keys. The orientation of the azimuth and plunge of the rotation axis and the angle of rotation can be modified in 10° steps. The final data are saved in a file and the rotation angles are given on the screen. The split utility for fault–slip data sets has been described in more detail earlier (Krejci and Richter, 1991). We have not included statistical operations because this is out of the scope of our contribution, and we think that available programs do an excellent job with statistical methods. Any of the published statistical algorithms, however, can be incorporated (for example as another subprogram) into TEC, if needed.

EXAMPLES

We demonstrate the flexibility of the TEC programs on a few examples from different studies (Fig. 1) by using different printers/plotters, symbols, sizes, and projection types. Directional data of the principal axes of the anisotropy of magnetic susceptibility (AMS) from the Nufenen pass (Switzerland) are plotted into equal area stereograms (Fig. 1A). The susceptibility device saves the directional data in a TEC compatible format with the plot symbols for the principal susceptibility axes (Ellwood, Hrouda, and Wagner, 1988) defined in a third column of the data file. The program Pol.C is a subprogram of the program collection that is used with the AMS unit. Data are printed for a first analysis with the attached 9 pin printer and later printed with higher quality or imported for further modifications into a drafting program. The second example (Fig. 1B) shows more than 1400 schistosity planes, which were measured during mapping in the Brixen quartzphyllite area (Southern Alps). We adjusted the symbol size to the large amount of data and used a laser printer for the output. Great circles of a set of conjugate faults plotted in an equal angular net (Fig. 1C) can be used if Anderson's (1942) fault model is valid to determine the direction of the principal stress axes that caused faulting. We used a laser printer for this plot. Figure 1D is an example of a fault–slip data set plotted with the method of Hoeppener (1955) that shows the pole to the fault plane and the direction of movement of the hanging wall. Fault–slip data can alternatively (Fig. 1E) be plotted as the great circle of the fault plane with an arrow indicating the direction of slip of the hanging wall. This example is plotted on a HP plotter. The symmetrical rose diagram (Fig. 1F) from joints in the Bozen quartzporphyry (Southern Alps) was modified with a drafting program. The direction of the principal stresses can be interpreted from this type of diagram, with the maximum compressional stress in the direction of the acute angle and the extensional stress in direction of the obtuse angle. The final example (Fig. 1G) is a digitized map, that was combined with various stereographic projections of structural data from the Southern Alps. The individual stereograms were saved as picture (*.pic) files and imported into the drafting program that was used to digitize the map.

HARDWARE AND SOFTWARE REQUIREMENTS

The software runs on any IBM or compatible PC with almost any monochrome or color graphics adapter or printer/plotter. The program is written in the programming language C. Table 1 is a selected list of supported hardware. We use the Borland Turbo-C compiler (version 2.0) or Borland C++ (version 2.0

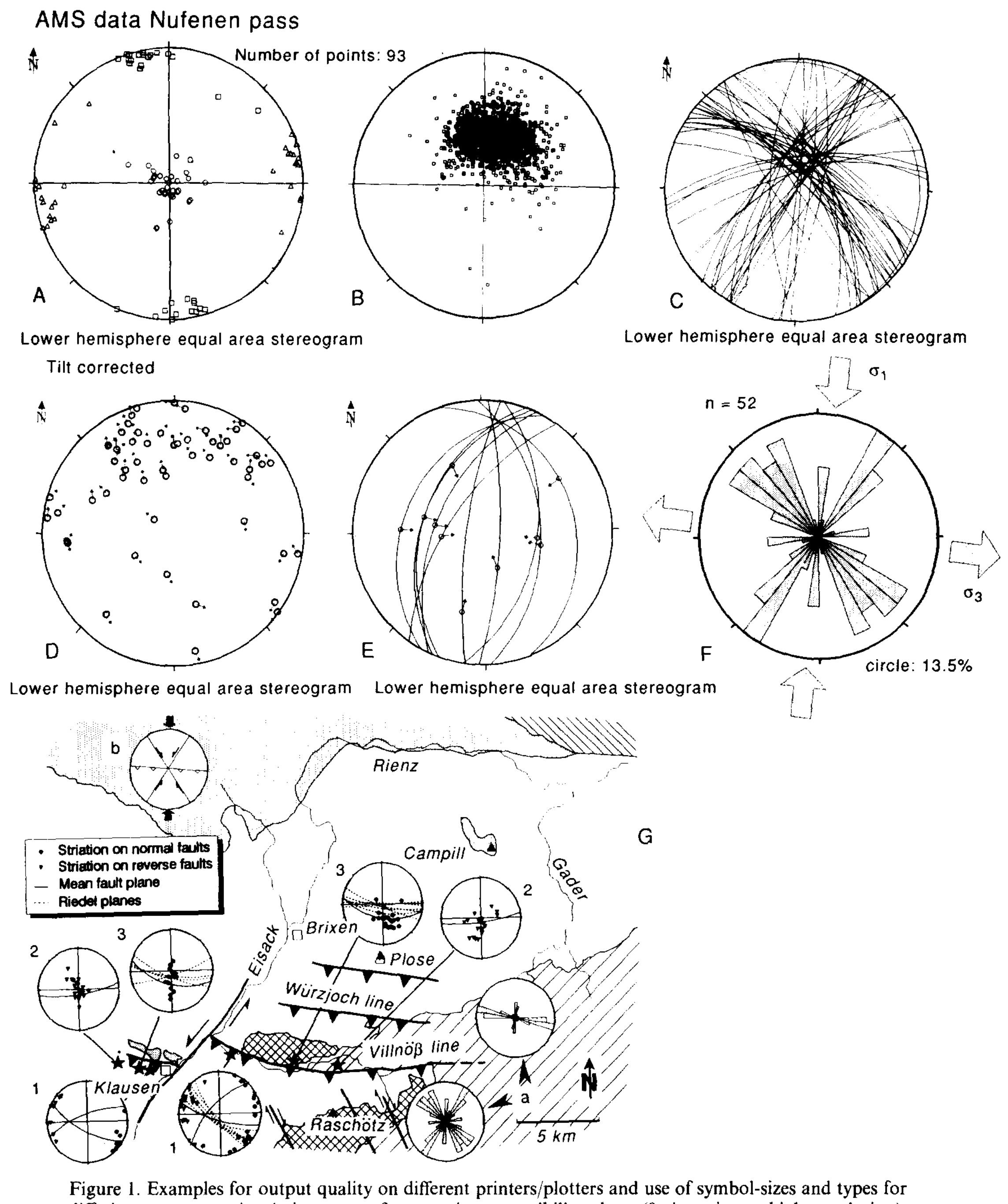

Figure 1. Examples for output quality on different printers/plotters and use of symbol-sizes and types for differing purposes. A—Anisotropy of magnetic susceptibility data (9 pin printer, high resolution); B—stereogram of 1400 schistosity poles (laser printer); C—great circles of 50 fault planes (laser printer); D—Hoeppener plot of fault slip data; E—Angelier plot of set of conjugate reverse faults (HP plotter); F—symmetric rose diagram of 50 faults; diagram has been modified and completed by drafting program; G—digitized map with fault-slip data and rose diagrams imported from TEC.

or 3.0). The graphic routines, installation to different graphic cards, and output handling use the toolbox GraphiC version 6.0 (Scientific Endeavors).

SUMMARY

We present a collection of different plotting programs to obtain prints of 3-D geological data. The main points that we are addressing are summarized in the following:

(1) The programs are virtually hardware independent and can be used to obtain a draft printout and publishable high-quality prints on almost every printer/plotter.
(2) Data files can be converted into graphic files that are compatible with professional drafting

programs. This opens a wide range of possibilities to work on the stereograms, for example modify or complete them, combine them with other draftings, or text.

(3) The programs can be modified easily for specific purposes, for example input file formats, default sizes or symbol stiles. Other plots can be included readily, for example by replacing the plot routine in any of the programs by another plot algorithm.

(4) The programs can be used under commandline, from a batch file, or from a menu overlay program that executes the individual programs. The default graphic settings can be modified in a definition file.

AVAILABILITY

The source code of the graphic programs, an executable version of the menu program TEC including editors to handle data and definition files, batch file management, and a short instruction can be obtained on 5.25 or 3.5 in. HD disks (DD disks upon request) by sending $20.- to one of the authors. Compilation of the graphic programs requires Turbo-C (Borland) and GraphiC (version 6.0, Scientific Endeavors, Kingston, Tennessee).

Acknowledgments—We are indebted specially to Harald Schmid (Tübingen) for his steady and sound advice through the years. Richter was funded partly by the Deutsche Forschungsgemeinschaft (DFG grant RI572/1-1). We appreciate the reviews and comments from two anonymous reviewers.

REFERENCES

Adam, J. F., 1989, Methoden und Algorithmen zur Verwaltung und Analyse axialer 3-D-Richtungsdaten und ihrer Belegungsdichten: Göttinger Arbeiten zur Geologie und Paläontologie, no. 40, 100 p.

Anderson, E. M., 1942, The dynamics of faulting (1st ed.): Oliver and Boyd, Edinburgh, 206 p.

Angelier, J., 1984, Tectonic analysis of fault slip data sets: Jour. Geophysical Res., v. 89, no. B7, p. 5835–5848.

Charlesworth, H., Cruden, D., Ramsden, J., and Huang, Q., 1989, ORIENT: an interactive FORTRAN-77 program for processing orientations on a microcomputer: Computers & Geosciences, v. 15, no. 3, p. 275–293.

Ellwood, B. B., Hrouda, F., and Wagner, J.-J., 1988, Symposia on magnetic fabrics: introductory comments: Phys. Earth Planet, Inter., v. 51, no. 4, p. 249–252.

van Everdingen, D. A., van Gool, J. A. M., and Vissers, R. L. M., 1992, QUICKPLOT: a microcomputer-based program for processing of orientation data: Computers & Geosciences, v. 18, no. 2/3, p. 183–287.

Griffis, R. A., Gustafson, S. J., and Adams, H. G., 1985, PETFAB: user considerate FORTRAN-77 program for the generation and statistical evaluation of fabric diagrams: Computers & Geosciences, v. 11, no. 4, p. 369–409.

Guth, P. L., 1987, MICRONET: interactive equal area and equal angle nets: Computers & Geosciences, v. 13, no. 5, p. 541–543.

Krejci, D., and Richter, C., 1991, SPLIT: a Turbo-C program for the graphical representation and separation of fault-slip data sets: Computers & Geosciences, v. 17, no. 6, p. 801–811.

Hoeppener, R., 1955, Tektonik im Schiefergebirge: Geol. Rundschau, v. 44, no. 1, p. 26–58.

Pecher, A., 1989, SCHMIDTMAC—a program to display and analyze directional data: Computers & Geosciences, v. 15, no. 8, p. 1315–1326.

Pilant, W. L., 1989, A PC-interactive stereonet plotting program: Computers & Geosciences, v. 15, no. 1, p. 43–58.

Vissers, R. J. M., and Bollegraaf, B., 1989, An algorithm for rotation of axial data: Computers & Geosciences, v. 15, no. 1, p. 157–161.

Wallbrecher, E., 1987, Gefügekundliche Arbeitsmethoden: Enke Verlag, Stuttgart, 208 p.

APPENDIX

GCir.C Listing

```
#include "graphic.h"
#include <dir.h>
#define PI 3.141592654
#define SIZE 2000
#define TRUE 1
#define FALSE 0

extern unsigned _stklen=6144;
extern int getch();

double (* azimut)[], (* dip)[];    int nitems;
char modelem[8], headline[80], bottomline[80], fl='Y', ptype[8];
char planar[]={"PL"}, ch='Y', wulff[]={"ANGULAR"}, lambert[]={"AREA"};
char drive[MAXDRIVE], dir[MAXDIR], fname[MAXFILE], ext[MAXEXT];
int flag;
/******************************************************************************/
int is_empty(char *s)
{ for(;*s;s++) if(!isspace(*s)) return FALSE; return TRUE; }
/******************************************************************************/
double rad2sin (double in)  { return(sin(in*PI/180));  }
double rad2cos (double in)  { return(cos(in*PI/180));  }
double rad2tan (double in)  { return(tan(in*PI/180));   }
double rad2atan (double in) { return(atan(in)*180/PI);  }
double deg2rad (double in)  { return(in*PI/180);        }
/***************************************************************************/
void readinput(char *inputfname, double azimut[], double dip[])
 { FILE *inputfile; char line[100]; int i;
   inputfile=fopen(inputfname,"rt");
   if (inputfile==NULL) { printf("Cannot open data file!\n");
                          printf("Press any key...\n"); getch(); exit(1); }
   fgets(line,100,inputfile); sscanf(line," %s",&modelem);
   for (i=0;!feof(inputfile) && fgets(line,100,inputfile)
                                 && i<SIZE;)
    { if(is_empty(line)) continue;
      sscanf(line," %lf %lf ",azimut+i,dip+i);
      i++;
    }
   fclose(inputfile); nitems=i;
 }
/***************************************************************************/
void readdef(char *inputfname)
 { FILE *inputfile; char line[100];
  inputfile=fopen(inputfname,"rt");
   if (inputfile==NULL) { sprintf(ptype,"%s","AREA"); }
    else
   { fgets(line,100,inputfile);   sscanf(line," %c",&fl);
   fgets(line,100,inputfile);   sscanf(line," %s",&ptype);
   fgets(line,100,inputfile);   sscanf(line,"%81[^\n]",&headline);
   fgets(line,100,inputfile);   sscanf(line,"%81[^\n]",&bottomline);
   fclose(inputfile); }
 }
/***************************************************************************/
double make_ticks()
{ int i;
  double tick_angle[]={0,45,90,135,180,225,270,315};
  for(i=0;i<8;i++)
   uline ( rad2sin(tick_angle[i])*50.0+50.0,
           50.0+rad2cos(tick_angle[i])*50.0,
           rad2sin(tick_angle[i])*52.0+50.0,
           50.0+rad2cos(tick_angle[i])*52.0,1 );
 return(0);
}
/***************************************************************************/
double plot_gcircles(double azimut[],double dip[])
{ int i; double dist, d, rad, radius_bc, xbc, ybc, begin_angle, end_angle;
  if(stricmp(ptype,wulff)==0)
  {
  for(i=0;i<nitems;i++)

 {
  if(dip[i] > 88.5)
   { uline ( rad2sin(azimut[i]+90)*50.0 + 50.0,
             50.0 + rad2cos(azimut[i]+90) * 50.0,
             rad2sin(azimut[i]+270)* 50.0 + 50.0,
             50.0+rad2cos(azimut[i]+270)*50.0, 1);
   }
   if(dip[i]<=88.5)
       { azimut[i]-=90;
         dist= 50.0 * rad2tan(dip[i]);
         rad = 50.0 * rad2tan((90-dip[i])/2);
         radius_bc=dist+rad;
         xbc= 50.0 + dist * rad2cos(azimut[i]-180);
         ybc= 50.0 - dist * rad2sin(azimut[i]-180);
         begin_angle =  270 - azimut[i] + dip[i];
         end_angle   =  360 - (azimut[i] - 90 + dip[i]);
         ucircle(xbc,ybc,radius_bc,deg2rad(begin_angle),
                                           deg2rad(end_angle) );
       }
  }  /* for */
 }  /* if */

if(stricmp(ptype,lambert)==0)
 {
for(i=0;i<nitems;i++)
 {
    if(dip[i] > 88.5 )
      {
       uline (    rad2sin(azimut[i]+90)*50.0 + 50.0,
                   50.0 + rad2cos(azimut[i]+90) * 50.0,
                   rad2sin(azimut[i]+270)* 50.0 + 50.0,
                   50.0+rad2cos(azimut[i]+270)*50.0, 1);
      }
  if(dip[i]<=88.5  && dip[i] > 0)
       {
           azimut[i]+=180;
           dist= ( 50.0/(2*sqrt(2))) * (rad2sin(dip[i])/
                                        (rad2sin((90-dip[i])/2)));
           rad = 50.0 * sqrt(2) * rad2sin((90-dip[i])/2);
           radius_bc=dist+rad;
           d= rad2atan (50.0/dist);
           xbc= 50.0 + dist * rad2cos(azimut[i]-90);
           ybc= 50.0 - dist * rad2sin(azimut[i]-90);
           begin_angle = 270-azimut[i]-d;
           end_angle   = 270-azimut[i]+d;
           ucircle(xbc,ybc,radius_bc,deg2rad(begin_angle),
                                      deg2rad(end_angle  ) );
       }
     }
    }
 return(0);
}
/****************************************************************************/
void free_storage(void)
 {  if (azimut) free(azimut); if (dip) free(dip);  }
void *qcalloc(size_t n,size_t size)
 {
  void *p=calloc(n,size); if(p) return(p);
   printf("Memory error!\n");
   printf("Press any key...\n"); getch(); exit(-1); return(0);
 }
/****************************************************************************/
void main(int argc,char *argv[])
{ char tkfn[80], def[80], name[80], header[80], bottom[80],items[80];
if (argc<2) {
printf("Usage is: <grc.exe> <work file name (*.*)>\n ");
printf("          Program search for def file <*.gro> \n");
printf("Output:    <tkf file name *.tkf> \n\n");
printf("Work file (three columns):\n");
printf("  First line define planar elments (use PLANAR)\n");
printf("  Followed by rows with two columns: 1. Azimut value (max. 2000)\n");
printf("                                     2. Dip value (max. 2000)\n\n");
printf("Definition file (4 rows):\n");
printf("  First line defines periphery print (Y or N)\n");
printf("  Second line defines projection type (ANGULAR or AREA)\n");
printf("  Following lines:              3. Headline\n");
printf("                                4. Bottomline\n\n");
        exit(-1); }
   atexit(free_storage);
   azimut=qcalloc(SIZE,sizeof(double));
   dip   =qcalloc(SIZE,sizeof(double));
    readinput(argv[1],(double *) azimut,(double *)dip);
     flag=fnsplit(argv[1],drive,dir,fname,ext);
      if (flag & FILENAME ) sprintf(name,"%s",fname);
        sprintf(def,"%s",name); strcat(def,".grc");
           readdef(def);  sprintf(tkfn,"%s",name); strcat(tkfn,".tkf");
     if(stricmp(modelem,planar)!=0)
     { printf("Error: No planar elements!\n");
       printf("Press any key...\n"); getch(); exit(-1);}
    sprintf(header,"\310%s",headline);
    sprintf(bottom,"\310%s",bottomline);
    sprintf(items,"\310Number of circles: %d ",nitems);
      bgnplot(1,'g',tkfn); startplot(7);
        font(2,"tec.fnt",'\310',"tecgrm.fnt",'\311',"",' ',"",' ');
        fillfont(1); metricunits(0);
         page(9.0,6.855); area2d(5.0,5.0);
         physor(1.0,1.0);  color(1);  axesoff(1);
         graf("",0.,10.,100.,"",0.,10.,100.);
         if (toupper(ch)==toupper(fl))
```

```
{
 lmargin(1.0); tmargin(6.2); ltline(bottom,0.2);
 lmargin(4.5); tmargin(0.7); ltline(items,0.1);
 tmargin(0.0); ltline(header,0.3);
 tmargin(6.0);
 if(stricmp(ptype,lambert)==0)
  prtfnt(3.5-strwidth("Lower hemisphere equal area stereogram",0.1),
        0.75,"Lower hemisphere equal area stereogram",0.1,0);
 if(stricmp(ptype,wulff)==0)
  prtfnt(3.5-strwidth("Lower hemisphere equal angular stereogram",0.1),
        0.75,"Lower hemisphere equal angular stereogram",0.1,0);
 vector(1.2,5.5,1.2,6.0,2.5,0.15,"11");
 lmargin(1.08); tmargin(.95); ltline("N",0.2);
}
make_ticks();
ucircle(50.0,50.0,50.0, 0, PI*2);
plot_gcircles((double *) azimut,(double *) dip);
endplot();
stopplot();
```