ISSN 0252-1075
Research Report No. RR-075

Contribution from

Indian Institute of Tropical Meteorology

SOME EXAMPLE OF X-Y PLOTS ON SILICON GRAPHICS

by

A.M. SELVAM
SUVARNA FADNAVIS
and
S.P. GHARGE

PUNE - 411 008
INDIA

MAY 1998

CONTENTS

Some Example of X-Y Plots on *Silicon Graphics*

A.M. Selvam, Suvarna Fadnavis and S.P.Gharge

Indian Institute of Tropical Meteorology, Pune 411 008

Abstract

Two-dimensional (X-Y) graphics plotting using *Silicon Graphics* is explained. Examples of programs using C Language for single and multiple graphics are illustrated with data.

Keywords : *Silicon Graphics*, Two-dimensional (X-Y) plots, C Language programming *Silicon Graphics*

1. Introduction

"*A picture is worth a thousand words.*" Basically the graphical idea is to plot on some sort of coordinate paper the various values of one quantity corresponding to the values of another . Graph serves as reassurance and a check of numerical analysis. Graphing facilitates the exposure of mistakes e.g. mistakes in calculation in tables of values may be easily shown by plotting the results.

Among the graphs that are used for display purposes primarily are line, pie, volumetric and histogram forms of graph.

Graphing frequently serves as useful technique to make a quick analysis of data, especially if one can utilize a straight-line format. The smoothness of nature is one of the features that make graphical technique most useful. Data smoothing, extrapolation into regions not measured, and interpolation between points that have actually been measured are all procedures most easily done with graphical analysis. It usually turns out that graphical methods are faster although not as accurate as other numerical techniques(Mayer, 1975).

In meteorology, data are huge and vast. To picture it, graph is the simplest solution. Several types of computer software for graphics are available, *Silicon Graphics* is one of such graphics platforms .In this paper some simple examples are given of X-Y graph plots using SILICON GRAPHICS IRIS graphics library. The IRIS GL is a library of subroutines for creating 2-D, 3-D graphics and animation. IRIS GL is implemented on SILICON GRAPHICS WORKSTATION (Power-Challenge).

Graphics development environment contains the tools and systems that you work with when you write GL programs. The tools available in the graphics development environment include

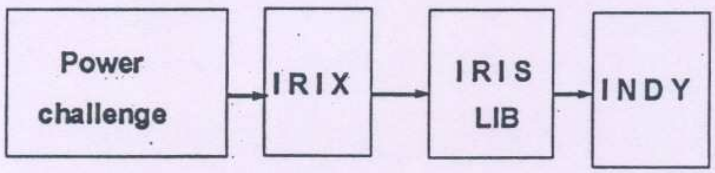
. Workstation
.The IRIX(5.3)operating system
.The IRIS graphics library

The system libraries include files, the X-window system and programming language that provides the interface to GL. Here "C" language is used for interface to GL because *font manager* is available in C Library only. Font manager enables the rotation of fonts. Fortran-77 can be used for interface to GL, but font manager is not available in Fortran-77. The IRIS power challenge workstation produces graphics operation through geometry pipeline. The geometry pipeline is like an assembly-line that performs the specialized graphics text that create and display graphics.

The IRIS operating system provides commands for setting-up and maintain system, creating and editing files and using IRIX system calls in GL applications. The GL is a library of subroutines that you call from a program written in C or another language to draw and animate 2-D and 3-D color graphics.

One can build an application using GL commands within the frame-work of the programming language. The programming language provides the logical structure for your program, and GL commands provide the interface to the graphics software and hardware. System libraries allow you to use capabilities such as graphics, fonts and math routines(Silicon Graphics,.1992).

2. Flow Diagram



3. Examples

Example I (**Appendix II**) shows simple X-Y plot for normalized standard deviation as X coordinate and relative dispersion as Y coordinate representing the measurement-scale relationship for variability of sea-level pressure data obtained from TOGA (*Tropical Ocean Global Atmosphere*) data for 1800 day period (January 1986-December 1990).The relevant technique of data analysis is discussed in a paper by Selvam and Sapre (1996). All available grid points (2.5 degree grid resolution) for 50N to 50S are used for study .A representative data set(taken from the data file **PRSLVL6**) for one grid point is shown in **Appendix I** where normalized relative dispersion (Y-coordinate)and normalized standard deviation (X-coordinate) are tabulated in seventh and eighth columns respectively. Program is written to read the X-Y values from the data file(**PRSLVL6**) and to plot the X-Y graphs.. In this example we are plotting data sets corresponding to 144 longitude locations for any one latitude. Multiple graphs can be plotted by repeating the procedure for the single graph . **Example II** (**Appendix III**) shows 21 latitudes, each containing 144 data sets (longitudes).

Each graphics package has its own advantages and disadvantages relating to the following requirements: (1) image resolution in pixels (2) flexibility regarding picture size, picture location, labeling, font size, format convertibility for different applications etc. (3) memory required for storing the image (4) speed of plotting (5) user friendliness. The package used in this work has also got the following advantages and disadvantages.

4. Advantages

(a) 2-D,3-D graphics and animation are provided. This graphics software is programmable and is more flexible than other graphics.

(b) Screen is defined in terms of pixels and user can move anywhere by defining pixel number. User can create large number of windows and can plot multiple plots. Pixel resolution is more and there is more clarity.

(c) Font library is available, wide range of fonts is available, rotation of font is also available.

(d) Wide range of conversion formats suitable for e-mail and other electronic transmission purposes is available. The picture images are portable to other systems and software environments.

5. Disadvantages

(a) As pixel resolution is more it occupies more memory space.

(b) User friendliness depends on familiarization with software package details and knowledge of "C" language used for programming.

6. Acknowledgement

The authors are grateful to Dr. A.S.R. Murty for his keen interest and encouragement during the course of the study. The authors acknowledge with thanks the constructive suggestions given by the Referee Dr. S.K.Behera ,Theoretical Studies Division, IITM. Thanks are due to Mrs.A.A.Shiralkar for the personal care she took to go through our manuscript and improve the

7. References

Meyer,S.I.,1975: Data Analysis For Scientists and Engineers, John wiley and sons, Inc., Canada, 49-50 pp.

Selvam, A.M., Sapre, V.V.,1996: Fractal Nature of Montblex Time Series Data, IITM research report ,Research report No. RR-069.

Silicon Graphics 1992: Graphics library programming guide Volume I Document number 007-1210-060,Silicon Graphics, Inc., Mountain View, California(This is available with Global Modeling Group, IITM.)

Spiegel M. R.,1961: Statistics , McGraw-Hill book Company, New York ,p359.

8. Appendix I

CD-ROM TOGA sea level pressure(mbs), 00gmt, January 1986 – December 1990 ,1800 days (daily values) at 2.5 degree grid intervals.

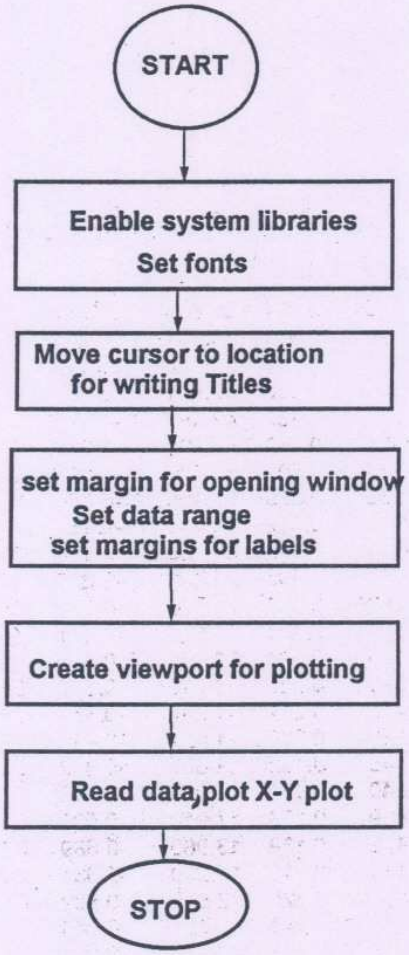
Latitude = 50.N Longitude = 0.0(Equator) Total Days = 1800

no	Av no	num	mean mb	sdev mb	rd(%)	norm rd(%)	t		D		D
1	1	1800	1016.944	10.099	0.993	100.000	0.719				
2	2	900	1016.945	9.530	0.937	94.362	0.560	0.084	1.084	-0.084	1.084
3	3	600	1016.945	9.083	0.893	89.940	0.467	0.097	1.097	-0.097	1.097
4	4	450	1016.945	8.611	0.847	85.263	0.401	0.115	1.115	-0.115	1.115
5	5	360	1016.945	8.256	0.812	81.747	0.350	0.125	1.125	-0.125	1.125
6	6	300	1016.945	7.915	0.778	78.31	0.308	0.136	1.136	-0.136	1.136
7	8	225	1016.945	7.420	0.730	73.44	0.242	0.148	1.148	-0.148	1.148
8	9	200	1016.945	7.319	0.720	72.467	0.215	0.147	1.147	-0.147	1.147
9	10	180	1016.945	7.040	0.692	69.709	0.191	0.157	1.157	-0.157	1.157
10	12	150	1016.945	6.207	0.610	61.454	0.149	0.196	1.196	-0.196	1.196
11	15	120	1016.945	6.040	0.594	59.809	0.098	0.190	1.190	-0.190	1.190
12	18	100	1016.945	5.480	0.539	54.261	0.056	0.212	1.212	-0.212	1.212
13	20	90	1016.945	5.493	0.540	54.392	0.032	0.203	1.203	-0.203	1.203
14	24	75	1016.945	4.929	0.485	48.801	-0.010	0.226	1.226	-0.226	1.226
15	25	72	1016.945	4.897	0.482	48.491	-0.019	0.225	1.225	-0.225	1.225
16	30	60	1016.945	4.597	0.452	45.521	-0.061	0.231	1.231	-0.231	1.231
17	36	50	1016.945	4.074	0.401	40.341	-0.103	0.253	1.253	-0.253	1.253
18	40	45	1016.945	3.853	0.379	38.155	-0.127	0.261	1.261	-0.261	1.261
19	45	40	1016.945	3.576	0.352	35.403	-0.154	0.273	1.273	-0.273	1.273
20	50	36	1016.945	3.810	0.375	37.729	-0.178	0.249	1.249	-0.249	1.249
21	60	30	1016.945	3.917	0.385	38.788	-0.220	0.231	1.231	-0.231	1.231
22	72	25	1016.945	3.593	0.353	35.576	-0.262	0.242	1.242	-0.242	1.242
23	75	24	1016.945	3.032	0.298	30.026	-0.271	0.279	1.279	-0.279	1.279
2	90	20	1016.945	2.686	0.264	26.591	-0.313	0.294	1.294	-0.294	1.294
25	100	18	1016.945	2.296	0.226	22.739	-0.337	0.322	1.322	-0.322	1.322
26	120	15	1016.945	2.142	0.211	21.207	-0.379	0.324	1.324	-0.324	1.324
27	150	12	1016.945	2.123	0.209	21.021	-0.430	0.311	1.311	-0.311	1.311
28	180	10	1016.945	1.849	0.182	18.303	-0.472	0.327	1.327	-0.327	1.327
29	200	9	1016.945	1.840	0.181	18.217	-0.496	0.321	1.321	-0.321	1.321
30	225	8	1016.945	1.809	0.178	17.910	-0.523	0.318	1.318	-0.318	1.318
31	300	6	1016.945	1.412	0.139	13.980	-0.589	0.345	1.345	-0.345	1.345
32	360	5	1016.945	1.115	0.110	11.039	-0.631	0.374	1.374	-0.374	1.374
33	450	4	1016.945	1.269	0.125	12.570	-0.682	0.339	1.339	-0.339	1.339
34	600	3	1016.945	0.502	0.049	4.975	-0.748	0.469	1.469	-0.469	1.469
35	900	2	1016.945	0.962	0.095	9.528	-0.841	0.346	1.346	-0.346	1.346

9. Appendix II

Example I

Flow chart



Program

```
/*Initialization of different libraries */
1.  #include <gl/gl.h>
2.  #include <gl/device.h>
3.  #include <fmclient.h>
4.  #include <stdio.h>
5.  #include <math.h>
6.  #include <ctype.h>
7.  # define a 0
8.  # define b 1
9.  # define ab 2
10. /*  program tgsvfrc*/
11. void main  (void)
12. {
13. short val;
14. fmfonthandle font1,font12,font13,font2,font22,font23,font3,font32,font33;
15. int i,ig,jdu, j ,k,nxl0,nxd,nolvl,ndxmx,ndymx,nxra,nyra,nrel,nrer,nreb,nret;
16. int nl1,nt,nxwid,nywid,ndx,ndy;
17. int m1,m2,mt,mb,ml,mr,mlat,llat,nlon,nlat,lat;
18. char str[150] ;
19. float xx,x[36], y[36],xmax,xmin,ymax,ymin,xmx,xmn,ymx,ymn,dxmx,dymx;
20. float vert[ab],xmxl,xmnr,ymxt,ymnb,xxmx,xxmn,xra,yra,xwid,ywid,xint,sint;
21. float d10, d11,d12,d13,d14,d15,d16,d17,d18,d19,andx,andy;
22. FILE *fp, *fopen();

23. fp = fopen("prslvl6", "r");

24. prefsize(1240,1000);
25. winopen(" ");
26. viewport(0,1240,0,1000);

27. qdevice(ESCKEY);
28. color(WHITE);
29. clear();
30. linewidth(2);
31. color(BLACK);
32. nrel=20;
33. nrer=1220;
34. nreb=10;
35. nret=990;
36. recti(nrel,nreb,nrer,nret);
37. /*select fonts for drawinf */
38. fminit();
39. font1 = fmfindfont("Times-Bold");
40. font12=fmscalefont(font1,20);
41. font13=fmscalefont(font1,18);
42. font2 = fmfindfont("Times-Roman");
43. font22=fmscalefont(font2,20);
```

```
font23=fmscalefont(font2,17);
font3 = fmfindfont("Times-BoldItalic");
font32=fmscalefont(font3,20);
font33=fmscalefont(font3,18);

fmsetfont(font12);
cmov2i(nrel+400,nret-50);
fmprstr(" FRACTAL DIMENSION");
cmov2i(nrel+125,nret-80);
fmprstr(" TOGA  00gmt daily mean sea level pressure (1800 days) jan
    1986 - dec 1990 ");
cmov2i(nrel+400,nret-105);
fmprstr(" 50N  to Equator  : 2.5 degree grid ");

cmov2i(nrel+300,nreb+77);
fmsetfont(font13);
fmprstr(" NORMALISED STANDARD DEVIATION  t  ");
fmrotatepagematrix(90);
cmov2i(nrel+80,nreb+260);
fmprstr("  NORMALISED RELATIVE DISPERSION ( % )  ");
cmov2i(95,150);
fmrotatepagematrix(-90);

fmsetfont(font33);
cmov2i(nrel+150,nreb+130);
fmprstr("  ");

cmov2i(nrel+150,nreb+100);
fmprstr("  ");
cmov2i(nrel+150,nreb+70);
fmprstr("  ");
cmov2i(nrel+150,nreb+40);
fmprstr("  ");

fmsetfont(font23);

/* left margin for graph and x y width in pixels for graph */
nxl0=180;
nxwid=800;
nywid=700;
xwid=nxwid;
ywid=nywid;

/*  data x y plot range */
xxmx = 3.;
xxmn = -1.;
xmx=xxmx;
xmn=xxmn;
ymx = 100.;
ymn = 0;
```

```
86. /* x y ranges */

87. xra=xmx-xmn;
88. yra=ymx-ymn;

89. /* pixel margins for x y labels */

90. ndx= 50;
91. ndy=50;
92. andx=ndx;
93. andy=ndy;
94. dxmx=(andx*xra)/xwid;
95. dymx=(andy*yra)/ywid;

    /*define top and left boundries */
96. nt=850;
97. nl1=nxl0;

98. xmxl=xmx;
99. xmnr=xmn;
100.   ymxt=ymx;
101.   ymnb=ymn;


102.   mt=nt;
103.   mb=mt-nywid;
104.
105.   ml=nl1;
106.   mr=ml+nxwid;
107.   ml=ml-ndx;
108.   xmxl=xmxl+dxmx;
109.   mt=mt+ndy;
110.   ymxt=ymxt+dymx;

111.   mr=mr+ndx;
112.   xmnr=xmnr-dxmx;
113.   ymnb=ymnb-dymx;
114.   mb=mb-ndy;

115.   viewport(ml,mr,mb,mt);

116.   nlon=144;

 /*read data from file */
117.
118.   fgets(str, 105, fp) ;
119.   printf("%s\n",str);

120.   mlat=1.;
121.   k=1;
```

```
122.

123.   while(k <= nlon)
124.   {

125.   if(k==1){
126.   fgets(str, 90, fp) ;
127.   printf("%s\n",str);
128.   fgets(str, 142, fp) ;
129.   printf("%s\n",str);
130.   fgets(str, 100, fp) ;
131.   printf("%s\n",str);
132.   }
133.   if(k>=2){
134.   fgets(str, 90, fp) ;
135.   printf("%s\n",str);
136.   fgets(str, 142, fp) ;
137.   printf("%s\n",str);
138.   fgets(str, 100, fp) ;
139.   printf("%s\n",str);
140.   fgets(str, 100, fp) ;
141.   printf("%s\n",str);
142.   }
143.   fscanf(fp,"%d %f %d %f %f %f %f %f
    ",&m1,&d10,&m2,&d11,&d12,&d13,&y[1],&x[1]);
144.   printf("%d %f %d %f %f %f %f %f\n
    ",m1,d10,m2,d11,d12,d13,y[1],x[1]);

145.   i=2;

146.   while( i <= 35)

147.   {
148.   fscanf(fp,"%d %f %d %f %f %f %f %f %f %f %f
    %f",&m1,&d10,&m2,&d11,&d12,&d13,&y[i],&x[i],&d16,&d17,&d18,&d19);
149.   printf("%d %f %d %f %f %f %f %f %f %f %f
    %f\n",m1,d10,m2,d11,d12,d13,y[i],x[i],d16,d17,d18,d19);
150.   i++ ;

151.   }

152.   drawit(35,x,y,xmxl,xmnr,ymxt,ymnb,xmx,xmn,ymn,ymx,xxmx,xxmn,font
    12,font23);
153.   k++;
154.   }

155.   qdevice(ESCKEY);

156.   while(1) {
```

```
158.   case ESCKEY:
159.   gexit();
160.   exit(0);
161.   break;
162.   }
163.   }

164.   }


165.   drawit(no,x,y,xmax,xmin,ymax,ymin,xmx,xmn,ymn,ymx,xxmx,xxmn,font
       12,font23)
166.   int no;
167.   float x[35],y[35],xmax,ymax,xmin,ymin,xmx,xmn,ymx,ymn,xxmx,xxmn;
168.   fmfonthandle font12,font23;

169.   {

170.   int i;
171.   float vert[ab],ysize,xsize,xpos,ypos,ypos1,ypos2,xra,yra,yint,xint;
172.   char dd[10];
173.   window(xmax,xmin,ymin,ymax,0.,0.);
174.   ortho2(xmax,xmin,ymin,ymax);
175.   rect(xmn,ymn,xmx,ymx);

176.   fmsetfont(font23);

177.   i = 1 ;
178.   bgnline();
179.   while (i <= no)
180.   {
181.   if(y[i] >= 1.) {
182.   vert[a]= x[i];
183.   vert[b]=y[i];
184.   v2f(vert);

185.   }
186.   i++ ;
187.

188.   }
189.   endline();
190.   yra=ymx-ymn;
191.   ysize=0.005*yra;
192.   yint=.1*yra;
193.   xra=xmx-xmn;
194.   xint=xra/8;
195.   xsize=xra/100;
196.   xpos=xsize;
197.   ypos=ymn-5.*ysize;
```

```
198.  ypos1=ymn;
199.  ypos2=ymx;


200.  drwticxx(xmx,xmn,xint,2);
201.  drwticyy(ymn,ymx,yint,xmx,3,xsize);
202.  drwticyy(ymn,ymx,yint,xmn,3,xsize);
203.  drwlbly(xmn-.1,ymn,ymx,yint);
204.  drwlbly(xmx+.2,ymn,ymx,yint);
205.  drwlblx(ymn-1.,xmx,xmn,xint,xpos);
206.  }

207.  drwlbly(sx,sbgn,send,sint)
208.  float sx,sbgn,send,sint;
209.  {
210.  float d;
211.  char dd[3];
212.  d=sbgn;
213.  while (d <= send)
214.  {
215.  cmov2(sx,d);
216.

217.  sprintf(dd,"%3.f", d ) ;
218.  fmprstr(dd);
219.  d=d+sint;
220.  }
221.  }

222.  drwlblx(sx,send,sbgn,sint,xpos)
223.  float sx,sbgn,send,sint,xpos;

224.  {
225.  float d;
226.  char dd[3];
227.  d=send;
228.  while (d >= sbgn)
229.  {
230.  cmov2(d,sx-50.*xpos);
231.
232.  sprintf(dd,"%4.1f", d ) ;
233.  fmprstr(dd);
234.  d=d-sint;
235.  }
236.  }


237.
238.  drwticxx(snd,sbgn,sint,m)
239.  int m;
```

```
240.  float sbgn,snd,sint;

241.  {
242.  int i,nn;
243.  float vert[3],p[2],q[2],s;
244.  q[1]=0.0;
245.
246.  nn=0;
247.  s=snd;
248.  while( s >= sbgn)
249.  {
250.  p[1]=s;
251.  p[2]=p[1];
252.  q[2]=1.5;
253.  nn=nn+1;
254.  if(  s == sbgn ) {
255.  q[2]=2.5;
256.  }
257.  if( nn == m) {
258.  q[2]= 2.5;
259.  nn = 1.;
260.  }

261.  i = 1 ;
262.  bgnline();
263.  while( i <= 2 )
264.  {
265.  vert[a]= p[i];
266.  vert[b]=q[i];
267.  v2f(vert);
268.  i++ ;
269.  }
270.  endline();

271.

272.  s=s-sint;
273.  }

274.  }

275.  drwticyy(sbgn,snd,sint,xp,m,sz)
276.  int m;
277.  float sbgn,snd,sint,xp,sz;

278.  {
279.  int i,nn;
280.  float vert[2],p[2],q[2],s;

281.  p[1]=xp;
```

```
282.  nn = 1;
283.  s = sbgn+sint;
284.  while ( s <= snd)
285.  {
286.
287.  q[0]=s;
288.  q[1]=s;
289.  if( xp >= 1. ) {
290.  p[0]=xp-sz;
291.  }
292.  if( xp < 1. ) {
293.  p[0]=xp+sz;
294.  }

295.  nn = nn + 1;

296.  if(nn == m) {

297.  if( xp >= 1. ) {
298.  p[0]=xp-2.*sz;
299.  }
300.  if( xp < 1. ) {
301.  p[0]=xp+2.*sz;
302.  }
303.  nn = 1;
304.  }
305.  if( s == sbgn) {
306.  p[0]=xp-2.*sz;
307.  }
308.
309.  i = 0 ;
310.  bgnline();
311.  while( i <= 1 )
312.  {
313.  vert[a]= p[i];
314.  vert[b]=q[i];
315.  v2f(vert);
316.  i++ ;
317.  }
318.  endline();
319.  s = s + sint;
320.  }

321.  }
```

Here, the first line **# include <gl/gl.h>** includes all the standard definitions for graphics. It must be included in every graphic program. Include files such as **math.h**, **device.h**, **stdio.h** provide definitions for system facilities that program uses. The subroutine **prefsize**(1240,1000) (line 24) tells the windows system that when a window is opened it should be 1240 pixels on X-axis and 1000 on Y-axis. It does not create a window , it establishes the initial size of root window to be opened. The call to **winopen**( ) (line25) actually creates the window and initializes the GL. Line on 28 (**colour** (white)) sets the back ground **colour** as white. The line numbers 30 and 31 sets line width and colour of graphic lines. Line 36 sets outline of rectangular region, **rect** draws an unfilled rectangle in X-Y plane with Z assumed to be zero.
Font (lines 39- 47) selects a raster font for drawing text string.
fmsetfont (line 48) sets the current font.
cmove (line50) renders a string in an current font.

Lines 74-78 set the left margin for graph then we set data range, X-Y range ,pixel margin for X-Y labels. Then we set the **viewport** allocates a rectangular area of the window for an image . Left , right, bottom, Top coordinates of the rectangular are set in lines 98-114.

Now whole background for drawing the picture is ready. Here, we are reading data from file **"PRSLVL6"** is reproduced in **Appendix I** for latitude 50N and longitude 0 degree as a representative sample. **drawit** (line 152) as a subroutine which actually draws the plot. In the **drawit** Subroutine we set the window . **ortho**(line 174) specifies a box shaped enclosure in the eye coordinate system that is mapped to the view port. Subroutine **drawticxx**, **drawticyy**, **drawlbly**, **drawlbx** sets X-ticks ,Y-ticks, Y-labels, X-labels. In this simple X-Y plot we are plotting sets corresponding to 144 longitude locations for any one latitude.

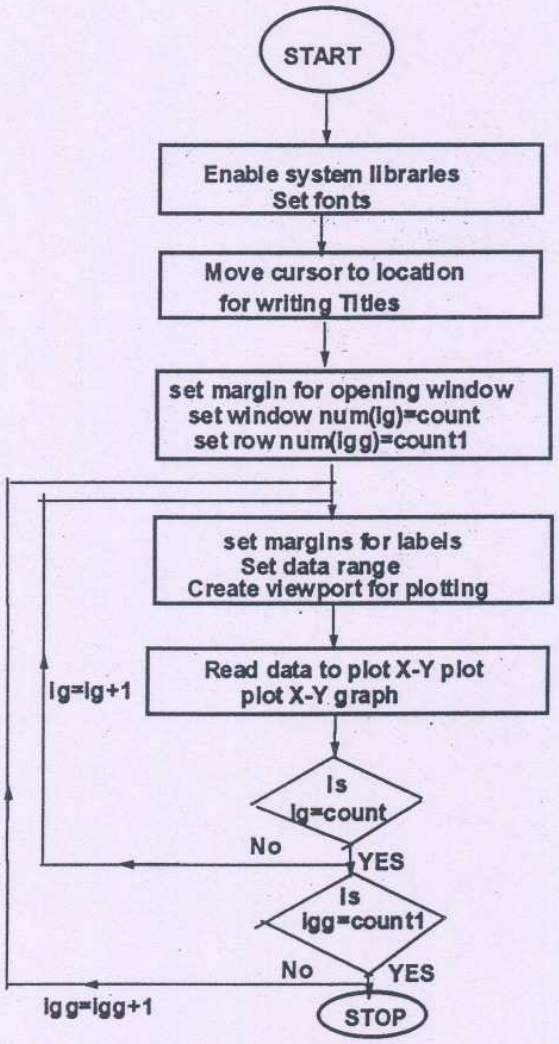
10. Appendix III

Multiple graphs can be plotted by repeating the procedure for a single graph.

Here is an example for 21 latitude, each containing 144 data set (longitudes).

Example II

Flow chart



Program

```
#include <gl/gl.h>
#include <gl/device.h>
#include <fmclient.h>
#include <stdio.h>
#include <math.h>
#include <ctype.h>
# define a 0
# define b 1
# define ab 2
/*  program tgsuafrc*/
void main  (void)
 {
    short val;
   fmfonthandle
font1,font12,font13,font2,font21,font22,font23,font3,font32,font33;
int i,ig,igg,jdu, j ,k,nxl0,nxd,nolvl,ndxmx,ndymx,nxra,nyra,nrel,nrer,nreb,nret;
int nl1,nt,nxwid,nywid,ndx,ndy,ntot;
int m1,m2,mt,mb,ml,mr,mlat,llat,nlon,nlat,lat;
float xx,xk[400],yk[400],x[36],y[36],xmax,xmin,ymax,ymin,xmx,xmn,
float ymx,ymn,dxmx,dymx;
float vert[ab],xmxl,xmnr,ymxt,ymnb,xxmx,xxmn,xra,yra,
float xwid,ywid,xint,sint,alo,ala;
float d10, d11,d12,d13,d14,d15,d16,d17,d18,d19,andx,andy;
char ttl11[20],ttl22[20],ttl33[20],ttl333[20];
char t1[20],t2[20],t3[20],t4[20],t5[20],t6[20],t7[20],t8[20],t9[20],t10[20];
char str[150] ,ttl1[20],ttl2[20],ttl3[20];
/* the cumulative statistical normal distribution tables is taken from Spigel
(1961) and is given below as tr[]*/
float tr[400][2] = {
        -.980,    16.350,
        -.970,    16.600,


.....
.....
        2.990,    99.860,
        3.000,    99.870,
};
   FILE *fp, *fopen();

 fp = fopen("prslvl6", "r");

  prefsize(1240,1000);
  winopen(" ");
  viewport(0,1240,0,1000);
```

```
 qdevice(ESCKEY);
 color(WHITE);
 clear();
 linewidth(1);
 color(BLACK);
 nrel=20;
 nrer=1220;
 nreb=10;
 nret=990;
recti(nrel,nreb,nrer,nret);
 fminit();
 font1 = fmfindfont("Times-Bold");
 font12=fmscalefont(font1,20);
 font13=fmscalefont(font1,18);
 font2 = fmfindfont("Times-Roman");
 font22=fmscalefont(font2,20);
 font21=fmscalefont(font2,14);
 font23=fmscalefont(font2,11);
 font3 = fmfindfont("Times-BoldItalic");
 font32=fmscalefont(font3,20);
 font33=fmscalefont(font3,18);

 fmsetfont(font12);
 cmov2i(nrel+400,nret-50);
/*  fmprstr(" FRACTAL DIMENSION");*/
 cmov2i(nrel+125,nret-60);
 fmprstr(" TOGA  00gmt daily mean sea level pressure (1800 days) jan 1986
- dec 1990 ");
 cmov2i(nrel+400,nret-95);
 fmprstr(" 50N  to Equator  : 2.5 degree grid ");

 cmov2i(nrel+300,nreb+80);
 fmsetfont(font13);
 fmprstr("NORMALISED STANDARD DEVIATION  t");
 cmov2i(nrel+150,nreb+40);
 fmsetfont(font33);
 fmprstr("Continuous lines --->Normalised relative dispersion(%)");
 cmov2i(nrel+175,nreb+10);
 fmsetfont(font33);
 fmprstr("    * * * * *    --->Statistical normal distribution ");
 fmrotatepagematrix(90);
 cmov2i(nrel+40,nreb+270);
 fmsetfont(font13);
 fmprstr("  NORMALISED RELATIVE DISPERSION (%)  ");
 cmov2i(95,150);
 fmrotatepagematrix(-90);

 fmsetfont(font33);
 cmov2i(nrel+150,nreb+130);
```

```
   cmov2i(nrel+150,nreb+100);
   fmprstr("  ");
   cmov2i(nrel+150,nreb+70);
   fmprstr("  ");
   cmov2i(nrel+150,nreb+40);
   fmprstr("  ");

   fmsetfont(font23);

/* left margin for graph and x y width in pixels for graph */
  nxwid=170;
  nywid=170;
  xwid=nxwid;
  ywid=nywid;

  nlon=144;

  fgets(str, 105, fp) ;
 nlat=22;
  mlat=1.;
ig=1;
while(mlat<=nlat){
ig=mlat;

if(ig==1||ig==6||ig==11||ig==17){
nxl0=100;
}

if(ig>=1 && ig<=5){
igg=1;
}
if(ig>=2 && ig<=5){
nxl0=nxl0+nxwid;
}
if(ig>=6 && ig<=10){
igg=2;
}
if(ig>=7 && ig<=10){
nxl0=nxl0+nxwid;
}
if(ig>=11 && ig<=16){
igg=3;
}
if(ig>=12 && ig<=16){
nxl0=nxl0+nxwid;
}
if(ig>=17 && ig<=22){
igg=4;
}
```

```
if(ig>=18 && ig<=22){
nxl0=nxl0+nxwid;
}
/*  data x y plot range */
     xxmx = 3.;
     xxmn = -1.;
     xmx=xxmx;
     xmn=xxmn;
     ymx = 100.;
     ymn = 0;
/* x y ranges */

    xra=xmx-xmn;
    yra=ymx-ymn;

/* pixel margins for x y labels */

  ndx= 50;
  ndy=50;
  andx=ndx;
  andy=ndy;
  dxmx=(andx*xra)/xwid;
  dymx=(andy*yra)/ywid;

nt=850-(igg-1)*nywid;
   nl1=nxl0;

xmxl=xmx;
   xmnr=xmn;
   ymxt=ymx;
   ymnb=ymn;

mt=nt;
  mb=mt-nywid;

  ml=nl1;
  mr=ml+nxwid;
  ml=ml-ndx;
  xmxl=xmxl+dxmx;
  mt=mt+ndy;
  ymxt=ymxt+dymx;


  mr=mr+ndx;
  xmnr=xmn-dxmx;
  ymnb=ymnb-dymx;
  mb=mb-ndy;

  viewport(ml,mr,mb,mt);
```

```
/*if(mlat<=21){*/
  k=1;

  while(k <= nlon)
{

   fscanf(fp,"%s %s %f %s %s %f %s %s %s
%d",ttl1,ttl11,&ala,ttl2,ttl22,&alo,ttl3,ttl33,ttl333,&ntot);

   fscanf(fp,"%s %s %s %s %s %s %s %s %s %s",t1,t2,t3,t4,t5,t6,t7,t8,t9,t10);
   fscanf(fp,"%s %s %s %s %s ",t1,t2,t3,t4,t5);

   fscanf(fp,"%d %f %d %f %f %f %f %f
",&m1,&d10,&m2,&d11,&d12,&d13,&y[1],&x[1]);

i=2;

  while( i <= 35)

{
   fscanf(fp,"%d %f %d %f %f %f %f %f %f %f %f
%f",&m1,&d10,&m2,&d11,&d12,&d13,&y[i],&x[i],&d16,&d17,&d18,&d19);
   i++ ;

   }
for(i=0;i<=394;i++){
xk[i]=tr[i][0];
yk[i]=tr[i][1];
}

if(mlat <= 21) {

drawit(ig,35,x,y,xmxl,xmnr,ymxt,ymnb,xmx,xmn,ymn,ymx,xxmx,xxmn,font12,f
ont23,font22,font21);
}
 k++;
 }


prstr(ig,399,xk,yk,xmxl,xmnr,ymxt,ymnb,xmx,xmn,ymn,ymx,xxmx,xxmn,font12
,font23,font22,font21);

if(mlat==22){

prstr(ig,399,xk,yk,xmxl,xmnr,ymxt,ymnb,xmx,xmn,ymn,ymx,xxmx,xxmn,font12
,font23,font22,font21);

}
mlat++;
}
```

```
    qdevice(ESCKEY);

    while(1) {
        switch(qread(&val)) {
        case ESCKEY:
   gexit();
            exit(0);
            break;
        }
    }
}

drawit(ig,no,x,y,xmax,xmin,ymax,ymin,xmx,xmn,ymn,ymx,xxmx,xxmn,font12,f
ont23,font22,font21)
 int no,ig;
 float x[35],y[35],xmax,ymax,xmin,ymin,xmx,xmn,ymx,ymn,xxmx,xxmn;
   fmfonthandle font12,font23,font22,font21;
{
int i;
float vert[ab],gk,ysize,xsize,xpos,ypos,ypos1,ypos2,xra,yra,yint,xint,aig;
 char dd[10];
  window(xmax,xmin,ymin,ymax,0.,0.);
  ortho2(xmax,xmin,ymin,ymax);
rect(xmn,ymn,xmx,ymx);

   fmsetfont(font23);
i = 1 ;
   bgnline();
   while (i <= no)
{
if(y[i] >= 0 && x[i]<=3.) {
   vert[a]= x[i];
   vert[b]=y[i];
   v2f(vert);
}
 cmov2(.2,90);
aig=ig;
 gk=50.-(ig-1.)*2.5;
     sprintf(dd,"%.1f", gk ) ;
  fmprstr(dd);
   i++ ;
}
   endline();


yra=ymx-ymn;
ysize=0.005*yra;
yint=.1*yra;
xra=xmx-xmn;
```

```
 xint=xra/4;
 xsize=xra/100;
 xpos=xsize;
 ypos=ymn-5.*ysize;
 ypos1=ymn;
 ypos2=ymx;


  drwticxx(xmx,xmn,xint,2);
if(ig>=17){
 drwlblx(ig,ymn,xmx,xmn,xint,xpos);
}
  drwticyy(ymn,ymx,yint,xmx,2,2.*xsize);
  drwticyy(ymn,ymx,yint,xmn,2,2.*xsize);

if (ig==1||ig==6||ig==11||ig==17){
 drwlbly(ig,xmx+16.*xpos,ymn,ymx,yint);
}
if (ig==5||ig==10||ig==16||ig==22){
 drwlbly(ig,xmn-xpos,ymn,ymx,yint);
}

}

prstr(ig,no,xk,yk,xmax,xmin,ymax,ymin,xmx,xmn,ymn,ymx,xxmx,xxmn,font12,f
ont23,font22,font21)
 int no,ig;
 float xk[400],yk[400],xmax,ymax,xmin,ymin,xmx,xmn,ymx,ymn,xxmx,xxmn;
   fmfonthandle font12,font23,font22,font21;
 {
int i;
float vert[ab],gk,ysize,xsize,xpos,ypos,ypos1,ypos2,xra,yra,yint,xint,aig;
  window(xmax,xmin,ymin,ymax,0.,0.);
  ortho2(xmax,xmin,ymin,ymax);
if (ig==22){
rect(xmn,ymn,xmx,ymx);
}

   fmsetfont(font21);
i=1;
while(i<=399){
 cmov2(xk[i],yk[i]-6.);
 fmprstr("*");
 i=i+20;
}
 yra=ymx-ymn;
 ysize=0.005*yra;
 yint=.1*yra;
 xra=xmx-xmn;
 xint=xra/4;
```

```
 xsize=xra/100;
 xpos=xsize;
 ypos=ymn-5.*ysize;
 ypos1=ymn;
 ypos2=ymx;

if(ig==22){
   fmsetfont(font23);
  drwticxx(xmx,xmn,xint,2);
if(ig>=17){
 drwlblx(ig,ymn,xmx,xmn,xint,xpos);
}
  drwticyy(ymn,ymx,yint,xmx,2,2.*xsize);
  drwticyy(ymn,ymx,yint,xmn,2,2.*xsize);

if (ig==1||ig==6||ig==11||ig==17){
 drwlbly(ig,xmx+16.*xpos,ymn,ymx,yint);
}
if (ig==5||ig==10||ig==16||ig==22){
 drwlbly(ig,xmn-xpos,ymn,ymx,yint);
}
}
}

 drwlbly(ig,sx,sbgn,send,sint)
float sx,sbgn,send,sint;
int ig;

{
 float d;
 char dd[3];
  d=sbgn;
if(ig==6||ig==11||ig==10||ig==15||ig==17||ig==22){
send = send - sint;
}
   while (d <= send)
{
 cmov2(sx,d);

     sprintf(dd,"%3.f", d ) ;
   fmprstr(dd);
  d=d+sint;
}
}

 drwlblx(ig,sx,send,sbgn,sint,xpos)
float sx,sbgn,send,sint,xpos;
int ig;
```

```
 float d;
int id;
 char dd[3];
if(ig==20||ig==17||ig==18||ig==19||ig==21){
sbgn=sbgn+sint;
}
  d=sbgn;
   while (d <= send)
{
 cmov2(d,sx-149.*xpos);
id=d;
      sprintf(dd,"%d", id ) ;
   fmprstr(dd);
  d=d+sint;
}
}


  drwticxx(snd,sbgn,sint,m)
  int m;
  float sbgn,snd,sint;

{
 int i,nn;
 float vert[3],p[2],q[2],s;
  q[1]=0.0;

  nn=0;

 s=snd;
 while( s >= sbgn)
{
  p[1]=s;
  p[2]=p[1];
  q[2]=1.5;
   nn=nn+1;
if(  s == sbgn ) {
    q[2]=2.5;
}
 if( nn == m) {
  q[2]= 2.5;
  nn = 1.;
}

  i = 1 ;
   bgnline();
   while( i <= 2 )
   {
   vert[a]= p[i];
   vert[b]=q[i];
```

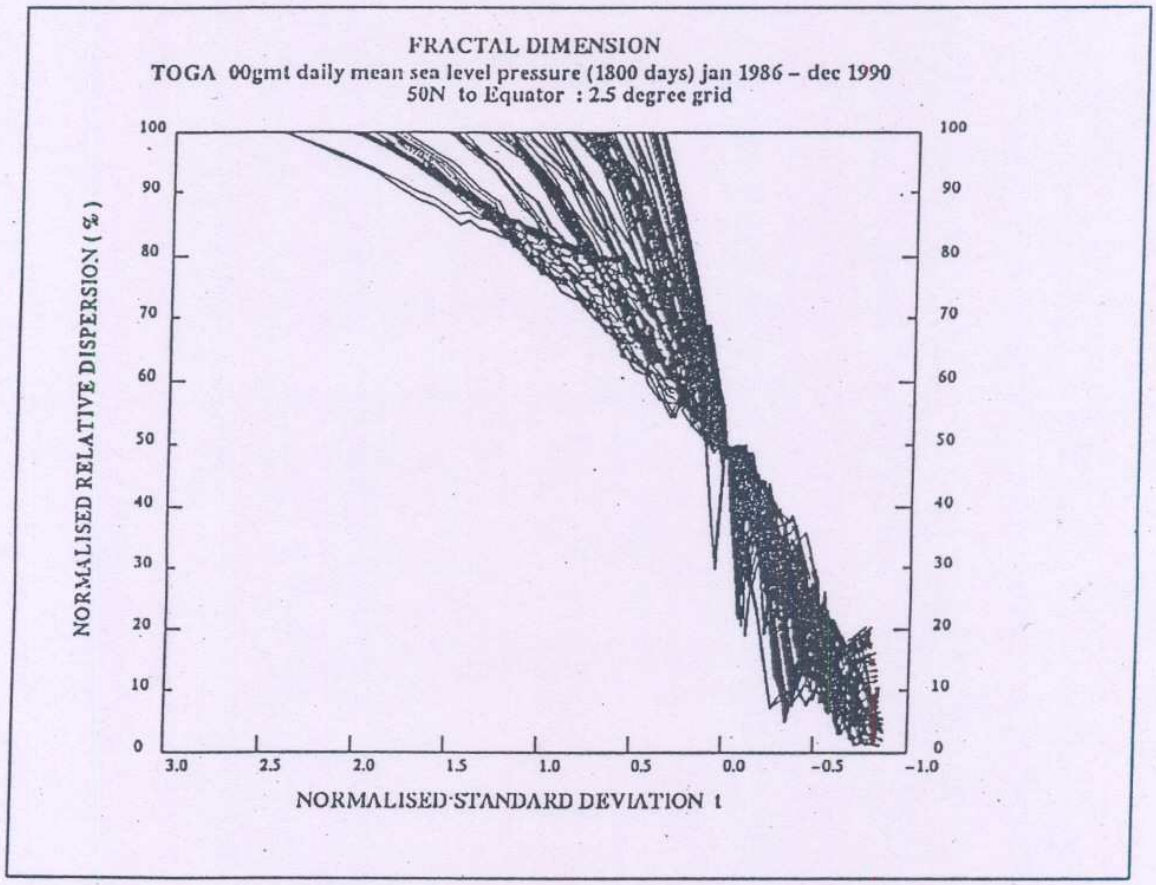
```
   v2f(vert);
   i++ ;
   }
   endline();

  s=s-sint;
}

 }


  drwlgticx(sbgn,snd,ysize,ypos,ymn,ymx)
  float sbgn,snd,ysize,ypos,ymn,ymx;

{
 int i,nn,m;
 float vert[3],p[2],q[2],s,g;
  float sint1, sint,size;
 int id;
 char dd[10];

  q[1]=ypos;
  g=1.0;

  if(ypos >= ymx) {
   ysize=-ysize;
}
  nn=0;

 s=sbgn;
 while( s <= snd)
{
 if(s<=9.){
 sint=1.;
 m=10;

}
if(s >= 10. && s <= 90.){
sint=10.;
m=10;
}
if(s >= 100. && s <= 900.){
sint=100.;
m=10;
}
if(s >= 1000. && s <= 2000.){
sint=500.;
m=10;
}
   size= 2 *ysize;
```

```
  p[1]=log10(s);
  p[2]=p[1];
  q[2]=q[1]+size;
   nn=nn+1;
if(  s= =sbgn){
q[2]=q[1]+2.*size;
}
if(nn= =m){
q[2]=q[1]+2.*size;
nn=1.
}
 i =1;
bgnline();
while(i<=2)
{
vert[a]=p[i];
vert[b]=q[i];
v2f(vert);
i++;
}
endline();

s=s-sint;
}
}


drwlgticx(sbgn,sent,ysize,ypose,ymn,ymx)
float sbgn,sent,ysize,ypose,ymn,ymx;
{
int i,nn,m;
float vert[3],p[2],q[2].s.g;
float sint1,sint,size;
int id;
char dd[10];
q[1]=ypose;
g= 1.0;
if(ypose>=ymn){
ysize= -ysize;
}
nn=0;
s=sbgn;
while(s<=send)
{
if(s<=9.){
sint=1.;
m=10;
}if(s>=10.&& s<=90.){
sint=100.;
m=10;
```

```
}
if(s>=1000. && s<= 2000.){
sint=500.;
m=10;
}size-2.*ysize;
p[1]=log10(s);
p[2]=p[1];
q[2]=q[1]+size;
nn=nn+1.;
if(s= = sbgn){
q[2]=q[1]+2.*size;
}
if(nn= =m){
q[2]=q[1]+2.*size;

nn=1.;
}

i=1;
bgnline();
while(I<=2)
{
vert[a]=p[i];
vert[b]=q[i];
v2f(vert);
i++;
}
endline();
s=s+sint;
}
}

drwticyy(sbgn,snd,sint,xp,m,sz)
int m;
float sbgn,snd,sint,xp,sz

{
int i, nn;
float vert[2],p[2],q[2],s;

p[1]=xp;
nn=1;
s=sbgn+sint;
while(s<=snd)
{

q[0]=s;
q[1]=s;
if(xp>=1.){
p[0]=xp-sz;
```

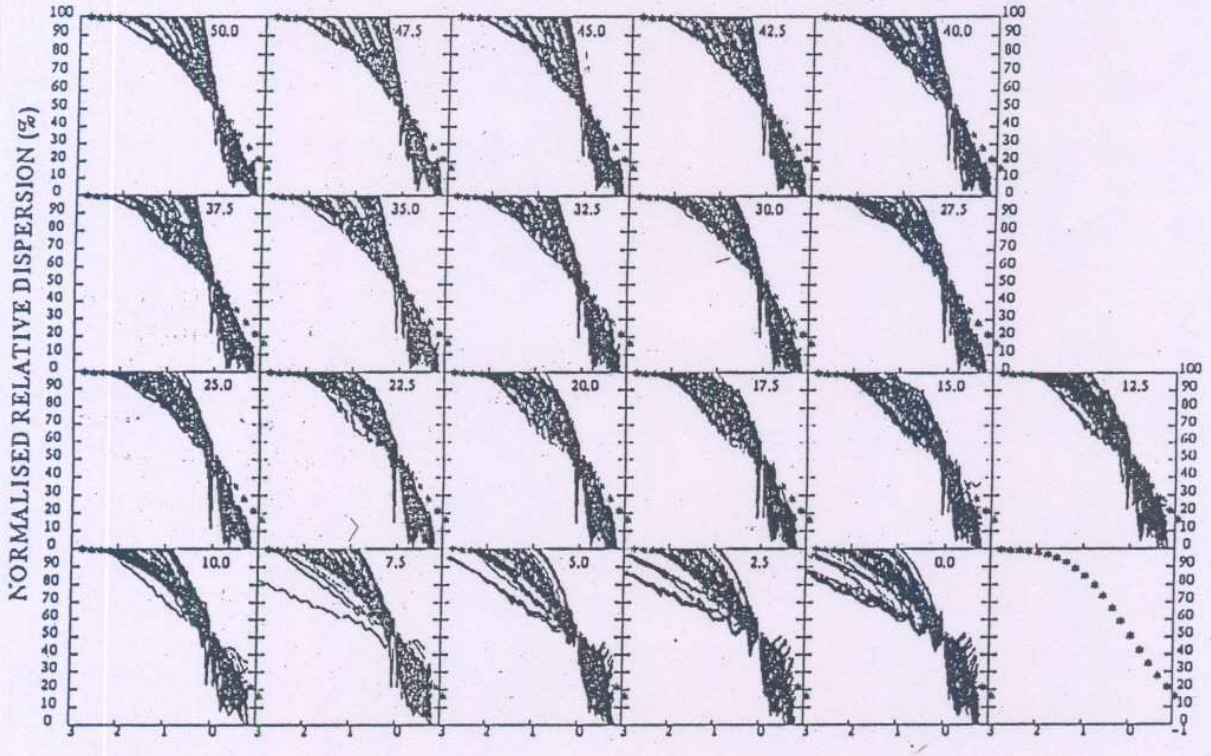
```
}
if(xp,1.){
p[0]=xp+sz;}
nn = nn + 1;
if(nn = = m) {
if( xp >= 1. ) {
  p[0]=xp-2.*sz;
}
  if( xp < 1. ) {
  p[0]=xp+2.*sz;
}
   nn = 1;
}
   if( s == sbgn) {
  p[0]=xp-2.*sz;
}

  i = 0 ;
   bgnline();
   while( i <= 1 )
   {
   vert[a]= p[i];
   vert[b]=q[i];
   v2f(vert);
   i++ ;
   }
   endline();
  s = s + sint;
}
}
```

In **Example II** normal distribution is plotted and is indicated by **star** marks. Data for normal distribution plot are read in the program, which is given at the start of the **Example II**. Subroutine **prstr** is used to plot normal distribution.


FRACTAL DIMENSION
TOGA 00gmt daily mean sea level pressure (1800 days) jan 1986 – dec 1990
50N to Equator : 2.5 degree grid
NORMALISED RELATIVE DISPERSION (%)
100
90
80
70
60
50
40
30
20
10
0
3.0
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
-1.0
NORMALISED STANDARD DEVIATION t

Example I

TOGA 00gmt daily mean sea level pressure (1800 days) jan 1986 – dec 1990
50N to Equator : 2.5 degree grid



Continuous lines --->Normalised relative dispersion(%)
***** --->*Statistical normal distribution*

Example II