GNU plotting utilities examples

Daisuke TOMINAGA
Computational Biology Reseach Center,
National Institute of Advanced Industrial Science and Technology, Japan
tominaga@cbrc.jp

July 31, 2009

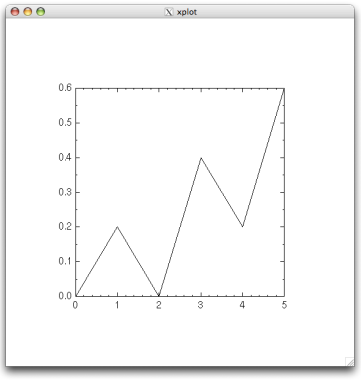
1 graph

1.1 General

(0)
```
cat > datafile
0.0 0.0
1.0 0.2
2.0 0.0
3.0 0.4
4.0 0.2
5.0 0.6^D
```

(1)
```
graph -T X < datafile
```

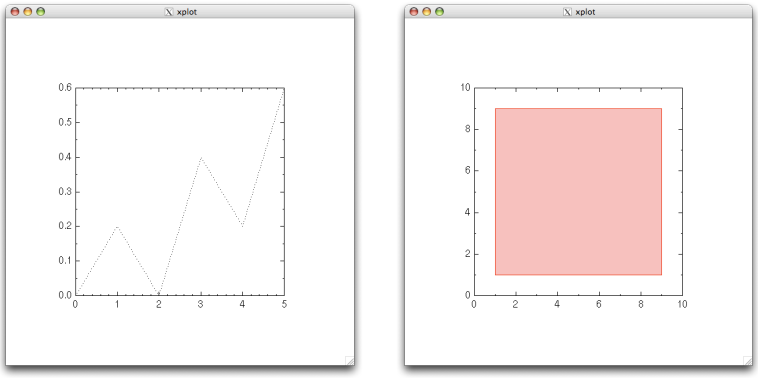
(1)



(1)
```
graph -T X -m 2 < datafile
```

(2)
```
echo 1 1 1 9 9 9 9 1 1 1 | graph -T X -C -m 1 -q 0.3
```

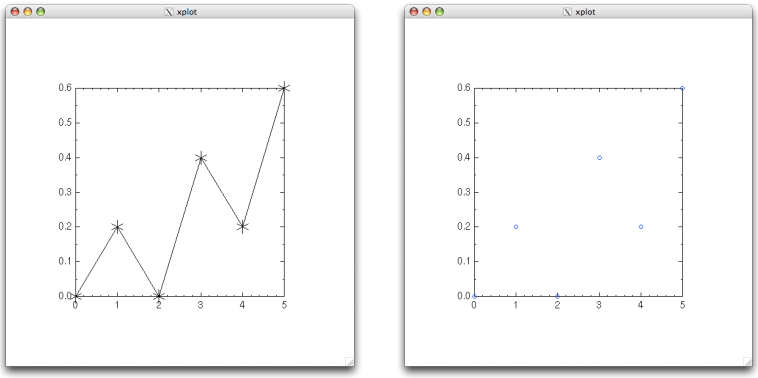
(1) (2)



(1) `graph -T X -S 3 0.1 < datafile`

(2) `graph -T X -C -m -3 -S 4 < datafile`

(1) (2)



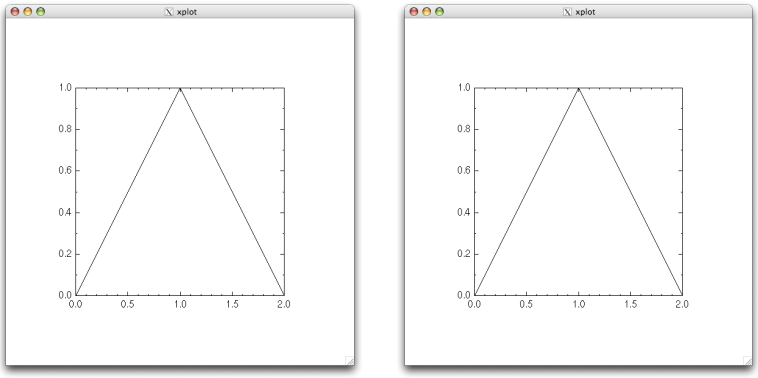
(1) `echo 0 1 0 | graph -T X -a`

(2) `echo 0 0 1 1 2 0 | graph -T X`

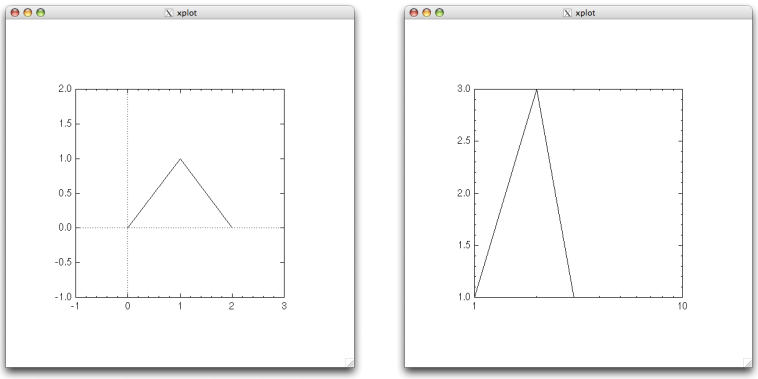
(3) `echo 0 0 1 1 2 0 | graph -T X -x -1 3 -y -1 2`

(4) `echo 1 1 2 3 3 1 | graph -T X -l x`
this command produces a warning message:
`graph: too few labelled axis ticks, adjust tick spacing manually`

(1) (2)



(3) (4)

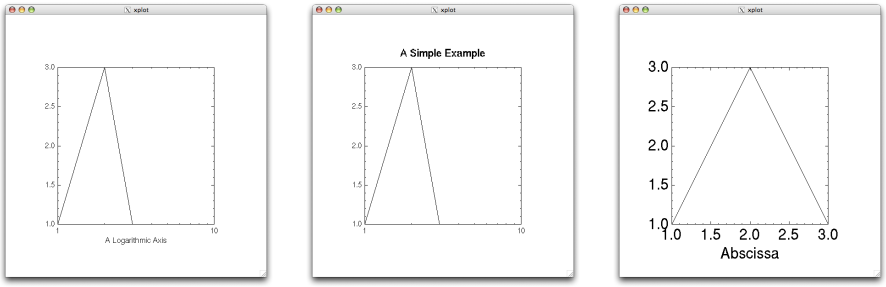


(1) `echo 1 1 2 3 3 1 | graph -T X -l x -X "A Logarithmic Axis"`

(2) `echo 1 1 2 3 3 1 | graph -T X -l x -L "A Simple Example"`
this command produces a warning message:
`graph: too few labelled axis ticks, adjust tick spacing manually`

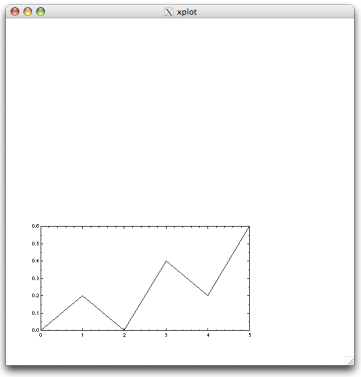
(3) `echo 1 1 2 3 3 1 | graph -T X -X "Abscissa" -f 0.1`

(1) (2) (3)



1.2 Resize

(0) `graph -T X -h .3 -w .6 -r .1 -u .1 < datafile`



1.3 Multiple datasets

(0) `cat > file1`
`0.0 0.0 1.0 0.2 2.0 0.0 3.0 0.4 4.0 0.2 5.0 0.6^D`
`cat > file2`
`0.2 0.2 1.2 0.4 2.2 0.2 3.2 0.6 4.2 0.4 5.2 0.8^D`
`cat > file3`
`0.5 0.5 1.5 0.7 2.5 0.5 3.5 0.9 4.5 0.7 5.5 1.0^D`

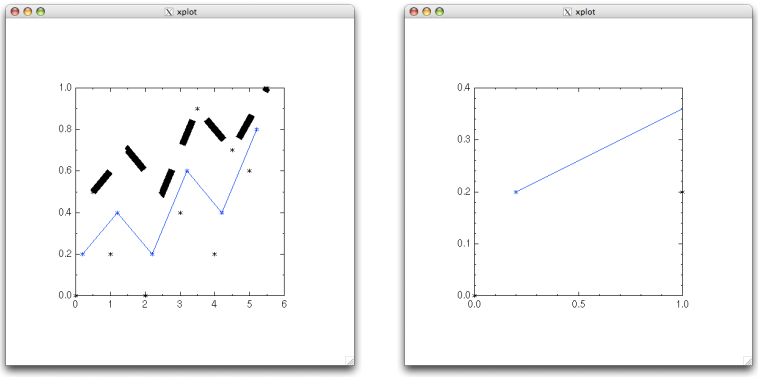
(1) `graph -T X -m 0 -S 3 file1 -C -m 3 file2 -C -W 0.02 file3`

(2) `graph -T X -x 0 1 0.5 -m 0 -S 3 file1 -C -m 3 file2`

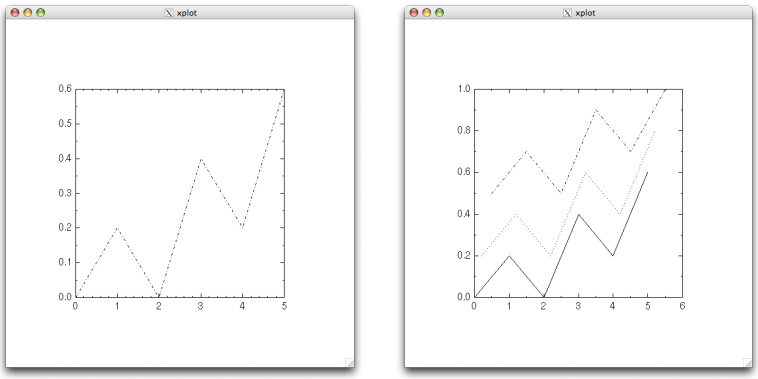
(3) `graph -T X -m 3 file1`

(4) `graph -T X file1 file2 file3`

(1) (2)



(3) (4)

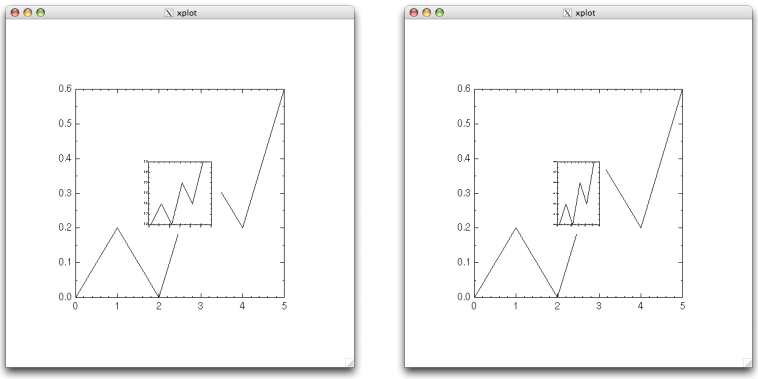


1.4 Multiplot

(1) `graph -T X file1 --reposition .35 .35 .3 file2`

(2) `graph -T X file1 --reposition .35 .35 .3 -w .4 -r .3 file2`

(1) (2)



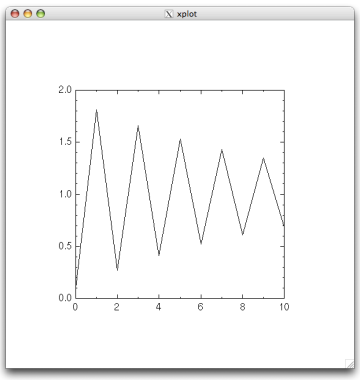
1.5 Binary data

(0)
```
(cat > df.c)<<INPUT
#include <stdio.h>
void main(void)
{
  float c = 1.0, x, y;
  for (x = 0; x <= 10; x+=1.0) {
    c *= -0.9; y = c + 1.0;
    fwrite(&x, sizeof (float), 1, stdout);
    fwrite(&y, sizeof (float), 1, stdout);
  }
}
INPUT
cc -o df df.c
./df > binary_datafile
```

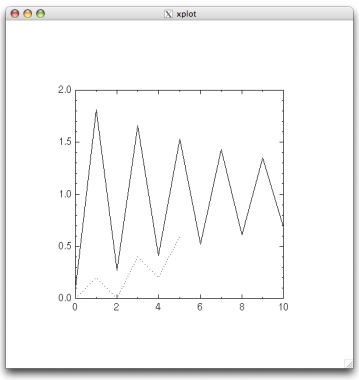
(1)
```
graph -T X -I f < binary_datafile
```

(2)
```
graph -T X -I f binary_datafile -I a file1
```

(1)



(2)

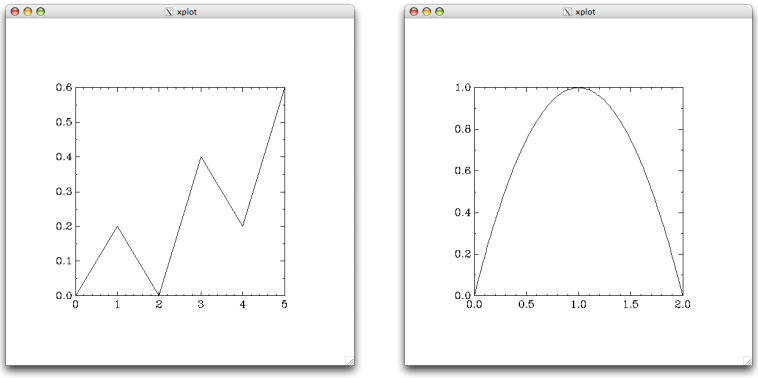


2 plot

(1)
```
cat > datafile
0.0 0.0 1.0 0.2 2.0 0.0 3.0 0.4 4.0 0.2 5.0 0.6^D
graph < datafile > test.meta
plot -T X test.meta
```

(2)
```
echo 0 0 1 1 2 0 | spline | graph > test.meta
plot -T X test.meta
```

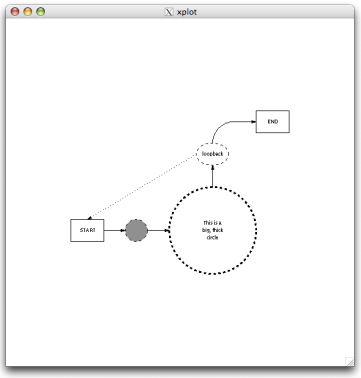
(1) (2)



3 pic2plot

(0)
```
(cat > test.pic)<<INPUT
.PS
box "START"; arrow; circle dashed filled; arrow
circle diam 2 thickness 3 "This is a" "big, thick" "circle" dashed;
up
arrow from top of last circle; ellipse "loopback" dashed
arrow dotted from left of last ellipse to top of last box
arc cw radius 1/2 from top of last ellipse; arrow
box "END"
.PE
INPUT
```

(1)
```
pic2plot -T X test.pic
```



4 plotfont

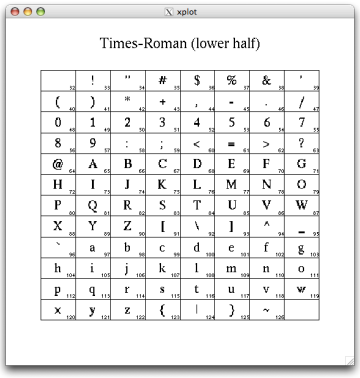
(0) `plotfont -T X --help-fonts`
This command outputs:

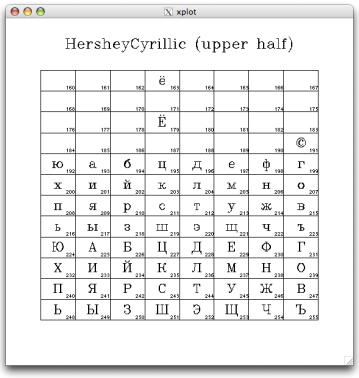
```
Names of supported Hershey vector fonts (case-insensitive):
    HersheySerif                        HersheyScript
    HersheySerif-Italic                 HersheyScript-Bold
    HersheySerif-Bold                   HersheyGothicEnglish
    HersheySerif-BoldItalic             HersheyGothicGerman
    HersheyCyrillic                     HersheyGothicItalian
    HersheyCyrillic-Oblique             HersheySerifSymbol
    HersheyEUC                          HersheySerifSymbol-Oblique
    HersheySans                         HersheySerifSymbol-Bold
    HersheySans-Oblique                 HersheySerifSymbol-BoldOblique
    HersheySans-Bold                    HersheySansSymbol
    HersheySans-BoldOblique             HersheySansSymbol-Oblique
Names of supported Postscript fonts (case-insensitive):
    Helvetica                           Bookman-Demi
    Helvetica-Oblique                   Bookman-DemiItalic
    Helvetica-Bold                      Courier
    Helvetica-BoldOblique               Courier-Oblique
    Helvetica-Narrow                    Courier-Bold
    Helvetica-Narrow-Oblique            Courier-BoldOblique
    Helvetica-Narrow-Bold               NewCenturySchlbk-Roman
    Helvetica-Narrow-BoldOblique        NewCenturySchlbk-Italic
    Times-Roman                         NewCenturySchlbk-Bold
    Times-Italic                        NewCenturySchlbk-BoldItalic
    Times-Bold                          Palatino-Roman
    Times-BoldItalic                    Palatino-Italic
    AvantGarde-Book                     Palatino-Bold
    AvantGarde-BookOblique              Palatino-BoldItalic
    AvantGarde-Demi                     ZapfChancery-MediumItalic
    AvantGarde-DemiOblique              ZapfDingbats
    Bookman-Light                       Symbol
    Bookman-LightItalic
Most core X Window System fonts, such as charter-medium-r-normal,
can also be used.
```

(1) `plotfont -T X Times-Roman`

(2) `plotfont -T X -2 HersheyCyrillic`

(1) (2)

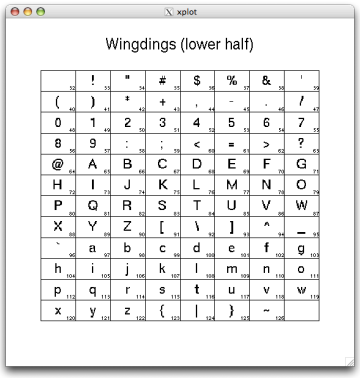


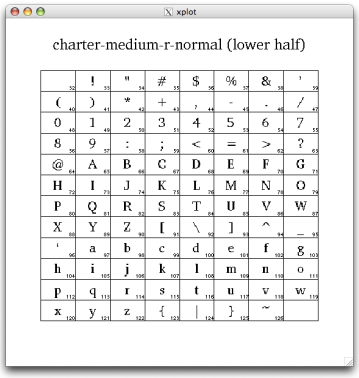


(1) `plotfont -T X Wingdings`
Substituted by the default font

(2) `plotfont -T X charter-medium-r-normal`

(1) (2)





5 spline

(0)
```
(cat > input)<<INPUT
0.0 0.0
1.0 1.0
2.0 0.0
INPUT
```

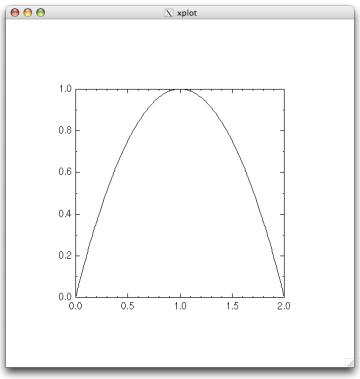
(1) `spline input | graph -T X`

(2) `echo 0 0 1 1 2 0 | spline | graph -T X`

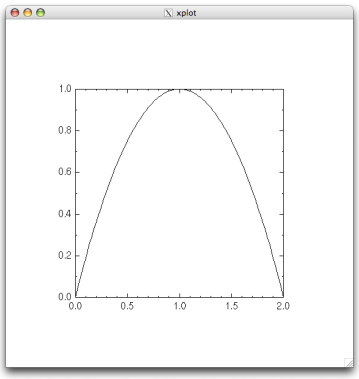
(3) `echo 0 0 1 1 2 0 | spline -n 20 -t 1.0 1.5 | graph -T X`

(4) `echo 0 0 1 0 2 0 | spline -T 10 | graph -T X`
this command produces a warning message:
`graph: separating identical upper and lower y limits`

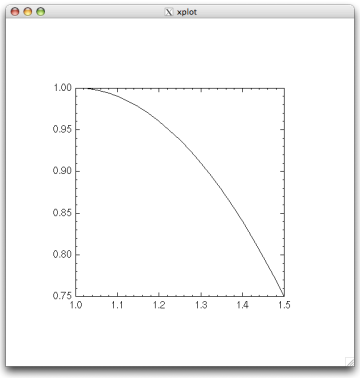
(1)



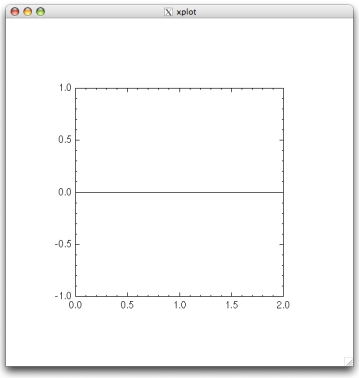
(2)



(3)



(4)

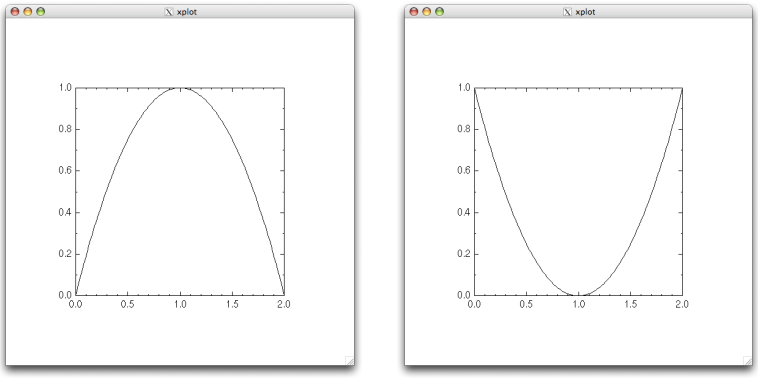


(0)
```
(cat > input2)<<INPUT
0.0 0.0 1.0
1.0 1.0 0.0
2.0 0.0 1.0
INPUT
```

(1) `spline -d 2 input2 | awk 'print $1,$2' | graph -T X`

(2) `spline -d 2 input2 | awk 'print $1,$3' | graph -T X`

(1) (2)

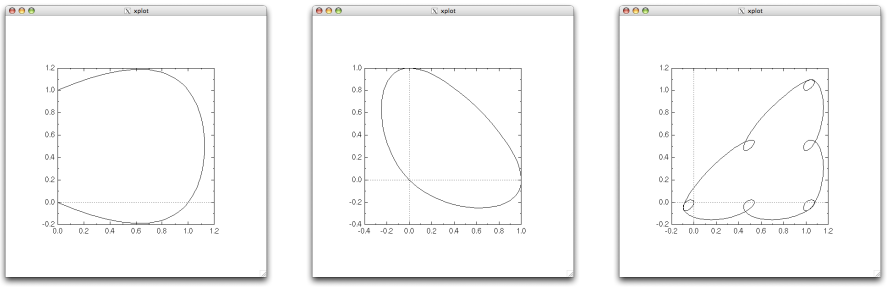


(1) `echo 0 0 1 0 1 1 0 1 | spline -d 2 -a -s | graph -T X`

(2) `echo 0 0 1 0 0 1 0 0 | spline -d 2 -a -s -p | graph -T X`

(3) `echo 0 0 1 0 1 1 0 0 | spline -d 2 -a -s -p -T -14 -n 500 | graph -T X`

(1) (2) (3)



6 ode

(1) `ode`
```
y' = y
y = 1
print t, y
step 0, 1
```
will output:
```
0 1
0.0625 1.064494
0.125 1.133148
0.1875 1.20623
0.25 1.284025
0.3125 1.366838
0.375 1.454991
0.4375 1.54883
0.5 1.648721
0.5625 1.755055
0.625 1.868246
0.6875 1.988737
0.75 2.117
0.8125 2.253535
0.875 2.398875
0.9375 2.553589
1 2.718282
```
(← blank line is output)
You can continue to input to `ode`.

(2)
```
step 1, 0
1 2.718282
0.9375 2.553589
0.875 2.398875
0.8125 2.253535
0.75 2.117
0.6875 1.988737
0.625 1.868246
0.5625 1.755055
0.5 1.648721
0.4375 1.54883
0.375 1.454991
0.3125 1.366838
0.25 1.284025
0.1875 1.20623
0.125 1.133148
0.0625 1.064494
0 1
```
(← blank line is output)
You can continue to input to `ode`.

(3)
```
step 1, 2
1 1
1.0625 1.064494
1.125 1.133148
1.1875 1.20623
1.25 1.284025
1.3125 1.366838
1.375 1.454991
1.4375 1.54883
1.5 1.648721
1.5625 1.755055
1.625 1.868246
1.6875 1.988737
1.75 2.117
1.8125 2.253535
1.875 2.398875
1.9375 2.553589
2 2.718282
```
(← blank line is output)
```
.
```

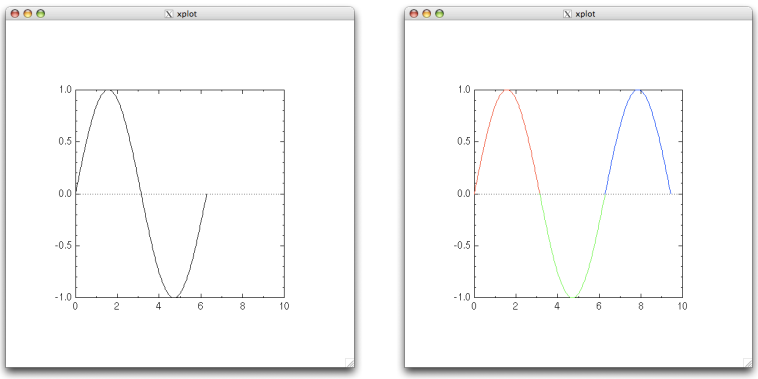
(4)
```
(cat > euler)<<INPUT
y = 1
y' = y
print t, y, y'
step 0, 1
INPUT
ode -f euler
0 1 1
0.0625 1.064494 1.064494
0.125 1.133148 1.133148
0.1875 1.20623 1.20623
0.25 1.284025 1.284025
0.3125 1.366838 1.366838
0.375 1.454991 1.454991
0.4375 1.54883 1.54883
0.5 1.648721 1.648721
0.5625 1.755055 1.755055
0.625 1.868246 1.868246
0.6875 1.988737 1.988737
0.75 2.117 2.117
0.8125 2.253535 2.253535
0.875 2.398875 2.398875
0.9375 2.553589 2.553589
1 2.718282 2.718282
```
(← blank line is output)

.

(1)
```
(cat > sine)<<INPUT
# sine :  y''(t) = -y(t), y(0) = 0, y'(0) = 1
sine' = cosine
cosine' = -sine
sine = 0
cosine = 1
print t, sine
INPUT
step 0, 2*PI" | ode -f sine | graph -T X -x 0 10 -y -1 1
```

(2)
```
(ode -f sine | graph -T X -C -x 0 10 -y -1 1) <<INPUT
step 0, PI
step PI, 2*PI
step 2*PI, 3*PI
.
INPUT
```

(1) (2)

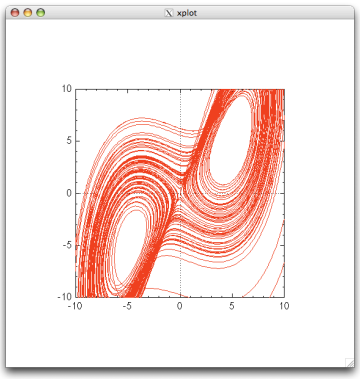


(1)
```
(cat > lorenz)<<INPUT # The Lorenz model, a system of three coupled
ODE's with parameter r.
x' = -3*(x-y)
y' = -x*z+r*x-y
z' = x*y-z
r = 26
x = 0; y = 1; z = 0
print x, y
step 0, 200
INPUT
ode < lorenz | graph -T X -C -x -10 10 -y -10 10
```

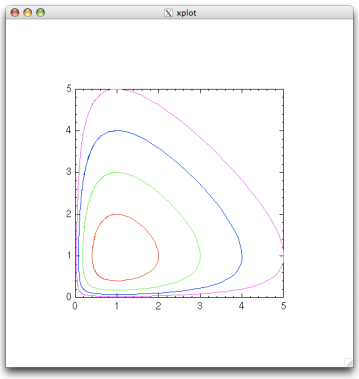
(2)
```
(ode | graph -T X -C -x 0 5 -y 0 5) <<INPUT
x' = (a - b*y) * x
y' = (c*x - d) * y
a = 1; b = 1; c = 1; d = 1;
```

```
print x, y
x = 1; y = 2
step 0, 10
x = 1; y = 3
step 0, 10
x = 1; y = 4
step 0, 10
x = 1; y = 5
step 0, 10
.
INPUT
```

(1)

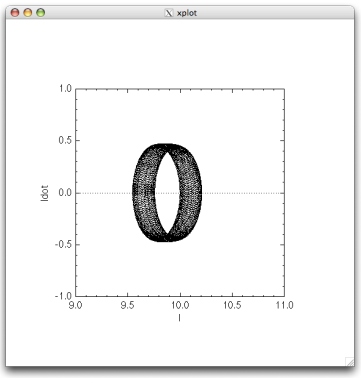


(2)



(1)
```
(cat > atwoods)<<INPUT m = 1
M = 1.0625
a = 0.5; adot = 0
l = 10; ldot = 0
ldot' = ( m * l * adot * adot - M * 9.8 + m * 9.8 * cos(a) ) /
(m + M)
l' = ldot
adot' = (-1/l) * (9.8 * sin(a) + 2 * adot * ldot)
a' = adot
print l, ldot
step 0, 400
INPUT
ode < atwoods | graph -T X -x 9 11 -y -1 1 -m 0 -S 1 -X l -Y ldot
```

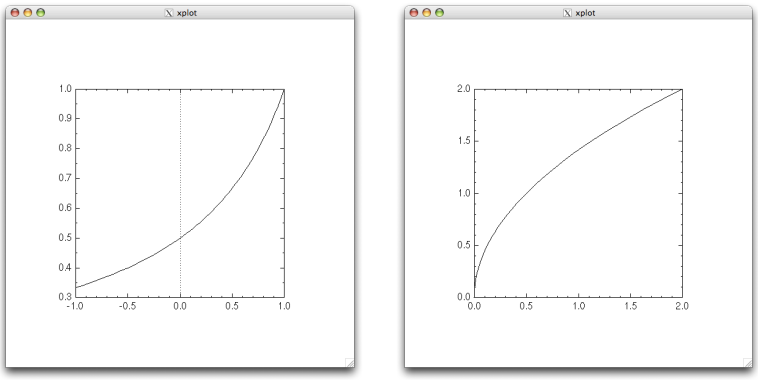
(1)



(1)
```
(ode |graph -T X) <<INPUT
y' = y^2
y = 1
step 1, -1
INPUT
```

(2)
```
(ode |graph -T X) <<INPUT
y' = 1/y
y = 2
step 2, -1
INPUT
```

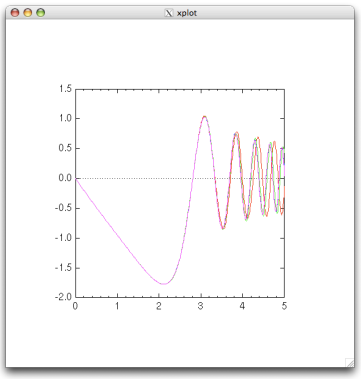
(1) (2)



(1)
```
(cat > qcd) <<INPUT
# an equation arising in QCD (quantum chromodynamics)
f' = fp
fp' = -f*g^2
g' = gp
gp' = g*f^2
f = 0; fp = -1; g = 1; gp = -1
print t, f
```

```
step 0, 5
^D
for sserr in "1" ".1" ".01" ".001"
do
ode -r $sserr < qcd
done | spline -n 500 | graph -T X -C
```

(1)

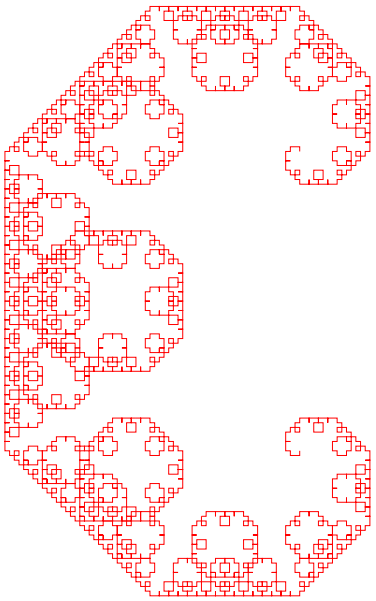


7 libplot

7.1 Sample drawings in C

```
#include <stdio.h>
#include <plot.h>
#define MAXORDER 12

void draw_c_curve (plPlotter *plotter, double dx, double dy, int order)
{
  if (order >= MAXORDER)
    /* continue path along (dx, dy) */
    pl_fcontrel_r (plotter, dx, dy);
  else
    {
      draw_c_curve (plotter,
                    0.5 * (dx - dy), 0.5 * (dx + dy), order + 1);
      draw_c_curve (plotter,
                    0.5 * (dx + dy), 0.5 * (dy - dx), order + 1);
    }
}

int main ()
```

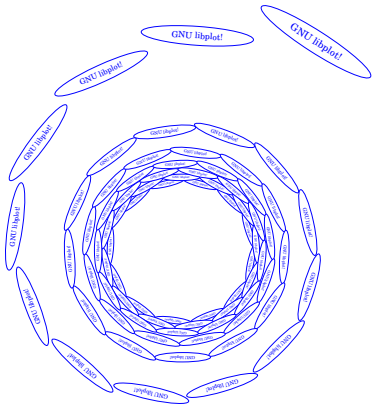
```
{
  plPlotter *plotter;
  plPlotterParams *plotter_params;

  /* set a Plotter parameter */
  plotter_params = pl_newplparams ();
  pl_setplparam (plotter_params, "PAGESIZE", "letter");

  /* create a Postscript Plotter that writes to standard output */
  if ((plotter = pl_newpl_r ("ps", stdin, stdout, stderr,
                             plotter_params)) == NULL)
    {
      fprintf (stderr, "Couldn't create Plotter\n");
      return 1;
    }

  if (pl_openpl_r (plotter) < 0)      /* open Plotter */
    {
      fprintf (stderr, "Couldn't open Plotter\n");
      return 1;
    }
  pl_fspace_r (plotter, 0.0, 0.0, 1000.0, 1000.0); /* set coor system */
  pl_flinewidth_r (plotter, 0.25);    /* set line thickness */
  pl_pencolorname_r (plotter, "red"); /* use red pen */
  pl_erase_r (plotter);               /* erase graphics display */
  pl_fmove_r (plotter, 600.0, 300.0); /* position the graphics cursor */
  draw_c_curve (plotter, 0.0, 400.0, 0);
  if (pl_closepl_r (plotter) < 0)     /* close Plotter */
    {
      fprintf (stderr, "Couldn't close Plotter\n");
      return 1;
    }

  if (pl_deletepl_r (plotter) < 0)    /* delete Plotter */
    {
      fprintf (stderr, "Couldn't delete Plotter\n");
      return 1;
    }
  return 0;
}
```



```
#include <stdio.h>
#include <plot.h>
#include <math.h>
#define SIZE 100.0   /* nominal size of user coordinate frame */
#define EXPAND 2.2   /* expansion factor for elliptical box */

void draw_boxed_string (plPlotter *plotter,
                        char *s, double size, double angle)
{
  double true_size, width;

  pl_ftextangle_r (plotter, angle);      /* set text angle (degrees) */
  true_size = pl_ffontsize_r (plotter, size);  /* set font size */
  width = pl_flabelwidth_r (plotter, s); /* compute width of string */
  pl_fellipserel_r (plotter, 0.0, 0.0,
                    EXPAND * 0.5 * width, EXPAND * 0.5 * true_size,
                    angle);              /* draw surrounding ellipse */
  pl_alabel_r (plotter, 'c', 'c', s);    /* draw centered text string */
}

int main()
{
  plPlotter *plotter;
  plPlotterParams *plotter_params;
  int i;
```

```
/* set a Plotter parameter */
plotter_params = pl_newplparams ();
pl_setplparam (plotter_params, "PAGESIZE", "letter");

/* create a Postscript Plotter that writes to standard output */
if ((plotter = pl_newpl_r ("ps", stdin, stdout, stderr,
                           plotter_params)) == NULL)
  {
    fprintf (stderr, "Couldn't create Plotter\");
    return 1;
  }

if (pl_openpl_r (plotter) < 0)      /* open Plotter */
  {
    fprintf (stderr, "Couldn't open Plotter\");
    return 1;
  }
/* specify user coor system */
pl_fspace_r (plotter, -(SIZE), -(SIZE), SIZE, SIZE);
pl_pencolorname_r (plotter, "blue");     /* use blue pen */
pl_fillcolorname_r (plotter, "white");   /* set white fill color */
pl_filltype_r (plotter, 1);   /* fill ellipses with fill color */
/* choose a Postscript font */
pl_fontname_r (plotter, "NewCenturySchlbk-Roman");

for (i = 80; i > 1; i--)      /* loop through angles */
  {
    double theta, radius;

    theta = 0.5 * (double)i;  /* theta is in radians */
    radius = SIZE / pow (theta, 0.35);  /* this yields a spiral */
    pl_fmove_r (plotter, radius * cos (theta), radius * sin (theta));
    draw_boxed_string (plotter, "GNU libplot!", 0.04 * radius,
                       (180.0 * theta / M_PI) - 90.0);
  }

if (pl_closepl_r (plotter) < 0)        /* close Plotter */
  {
    fprintf (stderr, "Couldn't close Plotter\");
    return 1;
  }
if (pl_deletepl_r (plotter) < 0)       /* delete Plotter */
  {
    fprintf (stderr, "Couldn't delete Plotter\");
    return 1;
  }
```

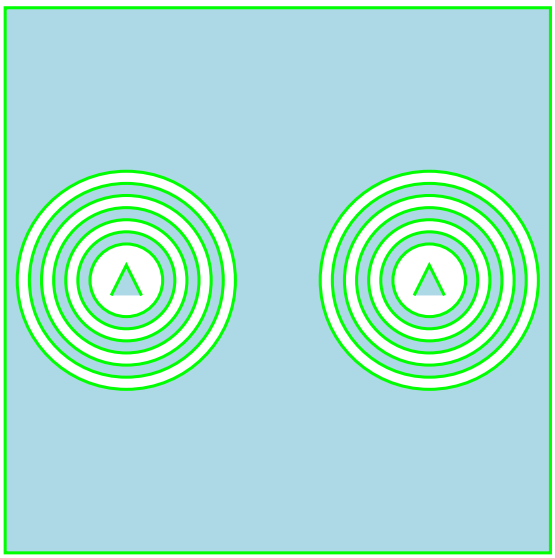
```
  return 0;
}
```



7.2 Simple paths and compound paths

```
#include <stdio.h>
#include <plot.h>

int main ()
{
  int i, j;
  plPlotter *plotter;
  plPlotterParams *plotter_params;

  /* set a Plotter parameter */
  plotter_params = pl_newplparams ();
  pl_setplparam (plotter_params, "PAGESIZE", "letter");
  /* create a Postscript Plotter that writes to standard output */
  plotter = pl_newpl_r ("ps", stdin, stdout, stderr, plotter_params);
  /* open Plotter, i.e. begin a page of graphics */
  pl_openpl_r (plotter);

  pl_fspace_r (plotter, 0.0, 0.0, 1000.0, 1000.0); /* set coor system */
  pl_flinewidth_r (plotter, 5.0);  /* set line thickness */
  pl_pencolorname_r (plotter, "green");
  pl_fillcolorname_r (plotter, "light blue");
  pl_filltype_r (plotter, 1);      /* do filling, full strength */
  pl_erase_r (plotter);            /* erase graphics display */

  /* draw a compound path consisting of 17 simple paths */

  /* draw the first simple path: a large box */
```

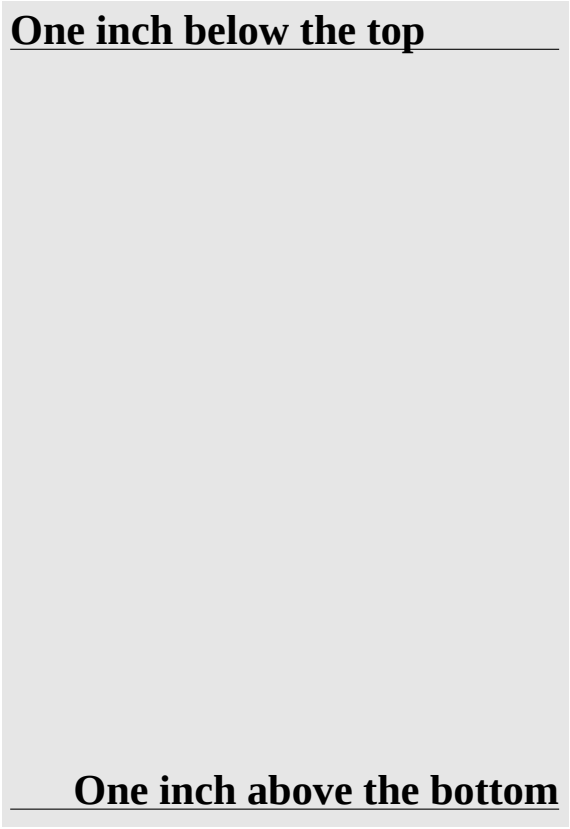
```
  pl_orientation_r (plotter, 1);
  pl_fbox_r (plotter, 50.0, 50.0, 950.0, 950.0);
  pl_endsubpath_r (plotter);
  for (i = 0; i < 2; i++)
    /* draw 8 simple paths that are nested inside the box */
    {
      /* first, draw 7 simple paths: nested circles */
      for (j = 9; j >= 3; j--)
        {
          pl_orientation_r (plotter, j           pl_fcircle_r (plotter, 250.0 + 500 * i, 5
          pl_endsubpath_r (plotter);
        }
      /* draw an open simple path comprising two line segments */
      pl_fmove_r (plotter, 225.0 + 500 * i, 475.0);
      pl_fcont_r (plotter, 250.0 + 500 * i, 525.0);
      pl_fcont_r (plotter, 275.0 + 500 * i, 475.0);
      pl_endsubpath_r (plotter);
    }
  /* formally end the compound path (not actually necessary) */
  pl_endpath_r (plotter);

  /* close Plotter, i.e. end page of graphics */
  pl_closepl_r (plotter);
  /* delete Plotter */
  if (pl_deletepl_r (plotter) < 0)
    {
      fprintf (stderr, "Couldn't delete Plotter\n");
      return 1;
    }
  return 0;
}
```



7.3 Drawing on a physical page

```
#include <stdio.h>
#include <plot.h>

int main()
{
  plPlotter *plotter;
  plPlotterParams *plotter_params;

  /* set page size parameter, including viewport size and location */
  plotter_params = pl_newplparams ();
  pl_setplparam (plotter_params, "PAGESIZE",
                 "letter,xsize=8.5in,ysize=11in,xorigin=0in,yorigin=0in");

  /* create a Postscript Plotter with the specified parameter */
  plotter = pl_newpl_r ("ps", stdin, stdout, stderr, plotter_params);

  pl_openpl_r (plotter);                /* begin page of graphics */
  pl_fspace_r (plotter,
               0.0, 0.0, 8.5, 11.0);   /* set user coor system */

  pl_fontname_r (plotter, "Times-Bold");
  pl_ffontsize_r (plotter, 0.5);        /* font size = 0.5in = 36pt */

  pl_fmove_r (plotter, 1.0, 10.0);
  pl_alabel_r (plotter, 'l', 'x', "One inch below the top");
```

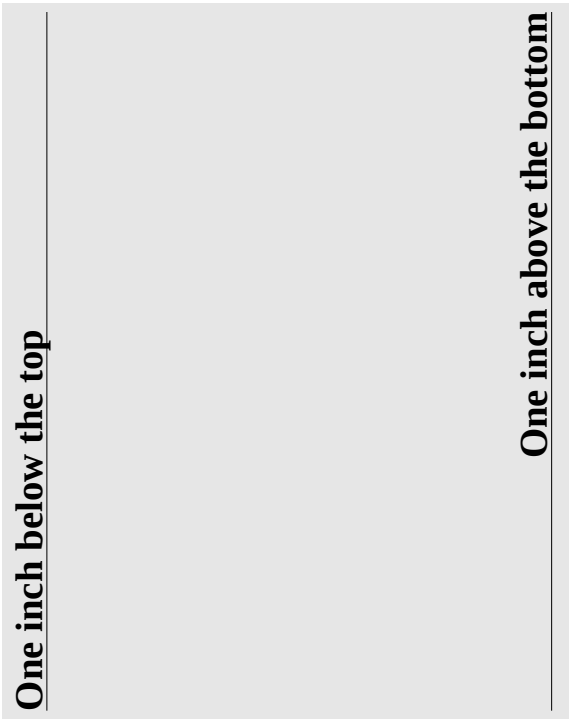
```
  pl_fline_r (plotter, 1.0, 10.0, 7.5, 10.0);

  pl_fmove_r (plotter, 7.5, 1.0);
  pl_alabel_r (plotter, 'r', 'x', "One inch above the bottom");
  pl_fline_r (plotter, 1.0, 1.0, 7.5, 1.0);

  pl_closepl_r (plotter);               /* end page of graphics */
  pl_deletepl_r (plotter);              /* delete Plotter */
  return 0;
}
```



(background is filled)

```
#include <stdio.h>
#include <plot.h>

int main()
{
  plPlotter *plotter;
  plPlotterParams *plotter_params;
```

```
  /* set Plotter parameters */
  plotter_params = pl_newplparams ();
  pl_setplparam (plotter_params, "PAGESIZE",
                 "letter,xsize=8.5in,ysize=11in,xorigin=0in,yorigin=0in");
  pl_setplparam (plotter_params, "ROTATION", "90");

  /* create a Postscript Plotter with the specified parameters */
  plotter = pl_newpl_r ("ps", stdin, stdout, stderr, plotter_params);

  pl_openpl_r (plotter);                /* begin page of graphics */
  pl_fspace_r (plotter,
               0.0, 0.0, 11.0, 8.5);   /* set user coor system */

  pl_fontname_r (plotter, "Times-Bold");
  pl_ffontsize_r (plotter, 0.5);        /* font size = 0.5in = 36pt */

  pl_fmove_r (plotter, 1.0, 7.5);
  pl_alabel_r (plotter, 'l', 'x', "One inch below the top");
  pl_fline_r (plotter, 1.0, 7.5, 10.0, 7.5);

  pl_fmove_r (plotter, 10.0, 1.0);
  pl_alabel_r (plotter, 'r', 'x', "One inch above the bottom");
  pl_fline_r (plotter, 1.0, 1.0, 10.0, 1.0);

  pl_closepl_r (plotter);               /* end page of graphics */
  pl_deletepl_r (plotter);              /* delete Plotter */
  return 0;
}
```



(background is filled)

7.4 Animated GIFs in C

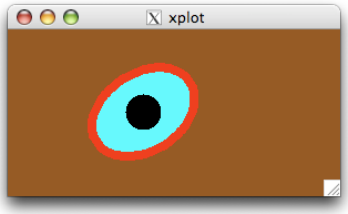
```
#include <stdio.h>
#include <plot.h>

int main()
{
  plPlotter *plotter;
  plPlotterParams *plotter_params;
  int i;

  /* set Plotter parameters */
  plotter_params = pl_newplparams ();
  pl_setplparam (plotter_params, "BITMAPSIZE", "150x100");
  pl_setplparam (plotter_params, "BG_COLOR", "orange");
  pl_setplparam (plotter_params, "TRANSPARENT_COLOR", "orange");
  pl_setplparam (plotter_params, "GIF_ITERATIONS", "100");
  pl_setplparam (plotter_params, "GIF_DELAY", "5");

  /* create a GIF Plotter with the specified parameters */
  plotter = pl_newpl_r ("gif", stdin, stdout, stderr, plotter_params);
```

```
  pl_openpl_r (plotter);                 /* begin page of graphics */
  pl_fspace_r (plotter,
               -0.5, -0.5, 149.5, 99.5); /* set user coor system */

  pl_pencolorname_r (plotter, "red");    /* use red pen */
  pl_linewidth_r (plotter, 5);           /* set line thickness */
  pl_filltype_r (plotter, 1);            /* objects will be filled */
  pl_fillcolorname_r (plotter, "black"); /* set the fill color */

  for (i = 0; i < 180 ; i += 15)
    {
      pl_erase_r (plotter);              /* begin new GIF image */
      pl_ellipse_r (plotter, 75, 50, 40, 20, i); /* draw an ellipse */
    }

  pl_closepl_r (plotter);                /* end page of graphics */
  pl_deletepl_r (plotter);               /* delete Plotter */
  return 0;
}
```

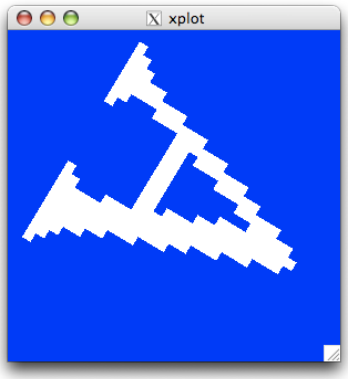
7.5 X Window System animations in C

```
#include <stdio.h>
#include <plot.h>

int main ()
{
  plPlotter *plotter;
  plPlotterParams *plotter_params;
  int i = 0, j;

  /* set Plotter parameters */
  plotter_params = pl_newplparams ();
  pl_setplparam (plotter_params, "BITMAPSIZE", "300x150");
  pl_setplparam (plotter_params, "VANISH_ON_DELETE", "yes");
  pl_setplparam (plotter_params, "USE_DOUBLE_BUFFERING", "yes");
```

```
  /* create an X Plotter with the specified parameters */
  if ((plotter = pl_newpl_r ("X", stdin, stdout, stderr,
                             plotter_params)) == NULL)
    {
      fprintf (stderr, "Couldn't create Plotter\n");
      return 1;
    }

  if (pl_openpl_r (plotter) < 0)         /* open Plotter */
    {
      fprintf (stderr, "Couldn't open Plotter\n");
      return 1;
    }
  pl_fspace_r (plotter,
               -0.5, -0.5, 299.5, 149.5);  /* set user coor system */
  pl_linewidth_r (plotter, 8);           /* set line thickness */
  pl_filltype_r (plotter, 1);            /* objects will be filled */
  pl_bgcolorname_r (plotter, "saddle brown"); /* set background color */
  for (j = 0; j < 300; j++)
    {
      pl_erase_r (plotter);                 /* erase window */
      pl_pencolorname_r (plotter, "red");   /* use red pen */
      pl_fillcolorname_r (plotter, "cyan"); /* use cyan filling */
      pl_ellipse_r (plotter, i, 75, 35, 50, i);  /* draw an ellipse */
      pl_colorname_r (plotter, "black"); /* use black pen and filling */
      pl_circle_r (plotter, i, 75, 12);  /* draw a circle [the pupil] */
      i = (i + 2) % 300;                 /* shift rightwards */
    }
  if (pl_closepl_r (plotter) < 0)        /* close Plotter */
    {
      fprintf (stderr, "Couldn't close Plotter\n");
      return 1;
    }

  if (pl_deletepl_r (plotter) < 0)       /* delete Plotter */
    {
      fprintf (stderr, "Couldn't delete Plotter\n");
      return 1;
    }
  return 0;
}
```




```
#include <stdio.h>
#include <plot.h>

int main()
{
  plPlotter *plotter;
  plPlotterParams *plotter_params;
  int angle = 0;

  /* set Plotter parameters */
  plotter_params = pl_newplparams ();
  pl_setplparam (plotter_params, "BITMAPSIZE", "300x300");
  pl_setplparam (plotter_params, "USE_DOUBLE_BUFFERING", "yes");
  pl_setplparam (plotter_params, "BG_COLOR", "blue");

  /* create an X Plotter with the specified parameters */
  plotter = pl_newpl_r ("X", stdin, stdout, stderr, plotter_params);

  /* open X Plotter, initialize coordinates, pen, and font */
  pl_openpl_r (plotter);
  pl_fspace_r (plotter, 0.0, 0.0, 1.0, 1.0);  /* use normalized coors */
  pl_pencolorname_r (plotter, "white");
  pl_ffontsize_r (plotter, 1.0);
  pl_fontname_r (plotter, "NewCenturySchlbk-Roman");

  pl_fmove_r (plotter, 0.5, 0.5);        /* move to center */
  while (1)                               /* loop endlessly */
    {
      pl_erase_r (plotter);
      pl_textangle_r (plotter, angle++); /* set new rotation angle */
      pl_alabel_r (plotter, 'c', 'c', "A"); /* draw a centered 'A' */
    }
  pl_closepl_r (plotter);                /* close Plotter */

  pl_deletepl_r (plotter);               /* delete Plotter */
  return 0;
}
```



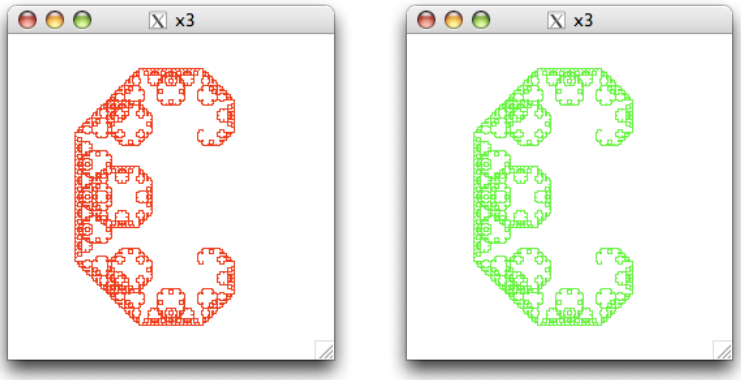
7.6 Advanced X Window System programming

```
#include <stdio.h>
#include <stdlib.h>
#include <plot.h>
#include <X11/Xlib.h>
#include <X11/Intrinsic.h>
#include <X11/Shell.h>
#include <X11/StringDefs.h>
#include <X11/Core.h>

plPlotter *plotter;
int green = 0;                  /* draw in green, not red? */

#define MAXORDER 12
void draw_c_curve (double dx, double dy, int order)
{
  if (order >= MAXORDER)
    /* continue path along (dx, dy) */
    pl_fcontrel_r (plotter, dx, dy);
  else
    {
      draw_c_curve (0.5 * (dx - dy), 0.5 * (dx + dy), order + 1);
      draw_c_curve (0.5 * (dx + dy), 0.5 * (dy - dx), order + 1);
    }
}

void Redraw (Widget w, XEvent *ev, String *params, Cardinal *n_params)
{
  /* draw C curve */
  pl_erase_r (plotter);
  pl_pencolorname_r (plotter, green ? "green" : "red");
```

```
  pl_fmove_r (plotter, 600.0, 300.0);
  draw_c_curve (0.0, 400.0, 0);
  pl_endpath_r (plotter);
}

void Toggle (Widget w, XEvent *ev, String *params, Cardinal *n_params)
{
  green = (green ? 0 : 1);
  Redraw (w, ev, params, n_params);
}

void Quit (Widget w, XEvent *ev, String *params, Cardinal *n_params)
{
  exit (0);
}

/* mapping of events to actions */
static const String translations =
"<Expose>:      redraw()\n
 <Btn1Down>:     toggle()\n
 <Key>q:         quit()";

/* mapping of actions to subroutines */
static XtActionsRec actions[] =
{
  {"redraw",            Redraw},
  {"toggle",            Toggle},
  {"quit",              Quit},
};

/* default parameters for widgets */
static String default_resources[] =
{
  "Example*geometry:      250x250",
  (String)NULL
};

int main (int argc, char *argv[])
{
  plPlotterParams *plotter_params;
  Arg wargs[10];                /* storage of widget args */
  Display *display;             /* X display */
  Widget shell, canvas;         /* toplevel widget; child */
  Window window;                /* child widget's window */
  XtAppContext app_con;         /* application context */
  int i;
```

```
char *bg_colorname = "white";

/* take background color from command line */
for (i = 0; i < argc - 1; i++)
  if (strcmp (argv[i], "-bg") == 0)
    bg_colorname = argv[i + 1];
/* create toplevel shell widget */
shell = XtAppInitialize (&app_con,
                         (String)"Example", /* app class */
                         NULL,              /* options */
                         (Cardinal)0,       /* num of options */
                         &argc,             /* command line */
                         argv,              /* command line */
                         default_resources,
                         NULL,              /* ArgList */
                         (Cardinal)0        /* num of Args */
                         );
/* set default widget parameters (including window size) */
XtAppSetFallbackResources (app_con, default_resources);
/* map actions to subroutines */
XtAppAddActions (app_con, actions, XtNumber (actions));
/* create canvas widget as child of shell widget; realize both */
XtSetArg(wargs[0], XtNargc, argc);
XtSetArg(wargs[1], XtNargv, argv);
canvas = XtCreateManagedWidget ((String)"", coreWidgetClass,
                                shell, wargs, (Cardinal)2);
XtRealizeWidget (shell);
/* for the canvas widget, map events to actions */
XtSetArg (wargs[0], XtNtranslations,
          XtParseTranslationTable (translations));
XtSetValues (canvas, wargs, (Cardinal)1);

/* initialize GNU libplot */
plotter_params = pl_newplparams ();
display = XtDisplay (canvas);
window = XtWindow (canvas);
pl_setplparam (plotter_params, "XDRAWABLE_DISPLAY", display);
pl_setplparam (plotter_params, "XDRAWABLE_DRAWABLE1", &window);
pl_setplparam (plotter_params, "BG_COLOR", bg_colorname);
plotter = pl_newpl_r ("Xdrawable", NULL, NULL, stderr,
                      plotter_params);
pl_openpl_r (plotter);
pl_fspace_r (plotter, 0.0, 0.0, 1000.0, 1000.0);
pl_flinewidth_r (plotter, 0.25);

/* transfer control to X Toolkit event loop (doesn't return) */
```

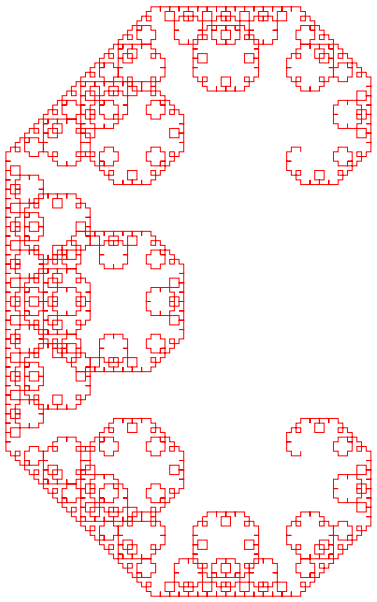
```
  XtAppMainLoop (app_con);

  return 1;
}
```



7.7 Sample drawings in C++

```
#include <plotter.h>
const int maxorder = 12;

void draw_c_curve (Plotter& plotter, double dx, double dy, int order)
{
  if (order >= maxorder)
    plotter.fcontrel (dx, dy); // continue path along (dx, dy)
  else
    {
      draw_c_curve (plotter,
                    0.5 * (dx - dy), 0.5 * (dx + dy), order + 1);
      draw_c_curve (plotter,
                    0.5 * (dx + dy), 0.5 * (dy - dx), order + 1);
    }
}

int main ()
{
  // set a Plotter parameter
  PlotterParams params;
  params.setplparam ("PAGESIZE", (char *)"letter");

  PSPlotter plotter(cin, cout, cerr, params); // declare Plotter
  if (plotter.openpl () < 0)                  // open Plotter
    {
      cerr << "Couldn't open Plotter\n";
```

```
      return 1;
    }

  plotter.fspace (0.0, 0.0, 1000.0, 1000.0); // specify user coor system
  plotter.flinewidth (0.25);       // line thickness in user coordinates
  plotter.pencolorname ("red");    // path will be drawn in red
  plotter.erase ();                // erase Plotter's graphics display
  plotter.fmove (600.0, 300.0);    // position the graphics cursor
  draw_c_curve (plotter, 0.0, 400.0, 0);
  if (plotter.closepl () < 0)      // close Plotter
    {
      cerr << "Couldn't close Plotter\n";
      return 1;
    }
  return 0;
}
```



8 Information

Contents of this document
All contents are examples shown in the reference manual of the GNU plotting utilities 2.5 and results produced by running these examples.

License of this document
Same as the reference manual of the GNU plotting utilities 2.5 (GFDL 1.2).

URL of this document
`http://www.cbrc.jp/%7Etominaga/translations/index.html#plotutils`
Japanese reference manual of the GNU plotting utilities is also available.

URL of the GNU plotting utilities
`http://www.gnu.org/software/plotutils/`

Test environment
All examples in this document are tested on GNU plotting utilities 2.5 with Xcode 3.1.3 on Mac OS X 10.5.7 (gcc 4.0.1 build by Apple Inc.) on MacBook (Core2Duo 2.4 GHz, 4 GB memory).

Typesetting
pTEX, Japanese version of TEX system by ASCII corp.

Contents