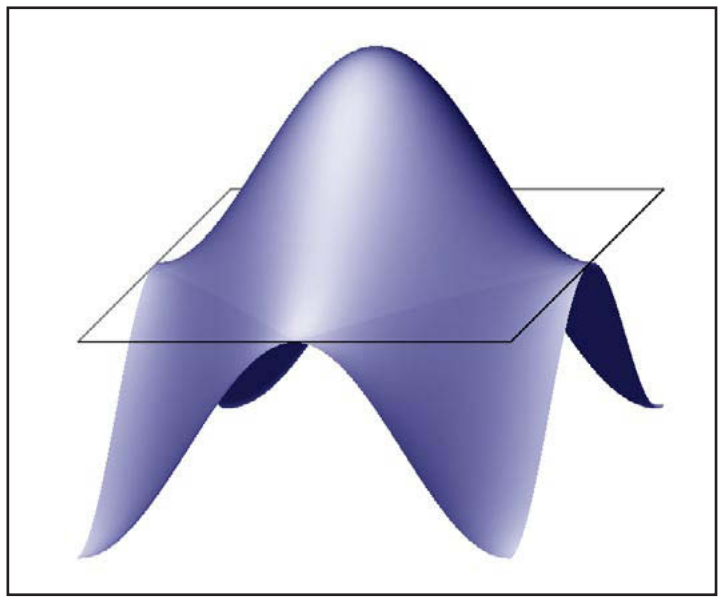
Gernot Hoffmann

PS-Tutor
Function Graphs
and Other Applications
for PostScript



Contents

1. Introduction

This document shows some examples for drawing smooth curves by PostScript.
The main purpose is the creation of mathematical function graphs for placing in the programs PageMaker and InDesign.
The CIE Chromaticity Diagram is an advanced example for accurate 2D-graphics, based on mathematical data.
New developments show that accurate 3D graphics are possible, though with many restrictions.
The programming techniques differ considerably from the typical PostScript style. PostScript uses the stack extensively. Optimized programs are hardly readable.
In these examples we use variables which are explicitly defined in the header and we assign new values also explicitly to named variables.
Obviously this is not professional PostScript programming, but the code is readable and

A very good PostScript editor/interpreter is essential.
We recommend PSAlter by Quite [4] . Unfortunately, PSAlter supports PostScript only up to level 2.

All 'local' variables in procedures *{ } bind def* require an initialization outside the procedure in advance to the first call. Otherwise they may collide with abbreviations in the host program. For example */r 0 def* prevents the interpreter from taking this as the abbreviation for 'rotate' in the host environment.
Newer programs do not use *bind* at all.

The matrix library chapter 18 is based on 'true' procedures by using local dictionaries instead of generally global variables.

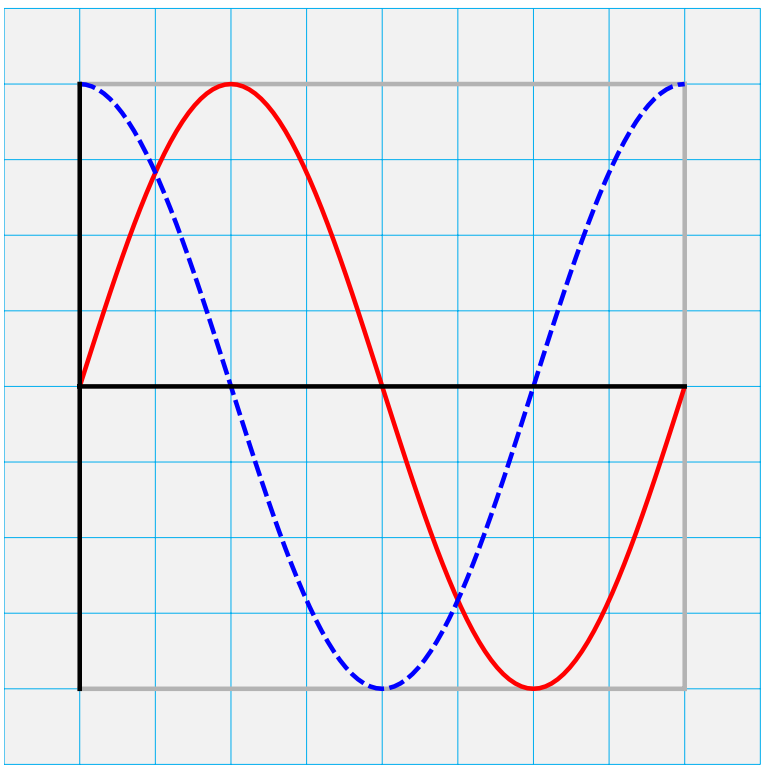
Even if a program seems to work perfectly in PSAlter, the placing in PageMaker may fail: only a gray box is shown. This is a result of a hidden error, like stack overflow in PM.
Syntax is essential - no commas for bounding box numbers.
PageMaker 6.52 cannot interpret *rectangle* operations.

For transfering PostScript files *.eps, it is recommended to rename them as *.txt. Otherwise it can happen that they are opened automatically by a PostScript interpreter like Photoshop.

Settings in Acrobat or Adobe Reader:
Edit > Preferences > Page display 72 dpi (instead of 96)
View document graphics by zoom=100% or 200%

2.1 Sine-Cosine

```
%!PS-Adobe-3.0       EPSF-3.0
%%BoundingBox:       0 0 284 284
%%Creator:           Gernot Hoffmann
%%Title:             Sine Curve
%%CreationDate:      Nov.29,2002

%—Variables

/mm {2.834646 mul} def    % points per mm

/n  100 def     % n steps for 0..1
/dx  1 n  div def

/sx  80 mm def       % length of x-axis,global scaling
/sy  40 mm def       % length of y-axis
/bx 100 mm def       % background box
/by 100 mm def

/x0  10 mm def       % origin of x-y-coordinates
/y0  50 mm def

/kx  1.00 def        % scale factors for x,y
/ky  0.50 def

/xa  x0 def          % start and end of axes
/xe  x0 sx add def

/ya  y0 sy sub def
/ye  y0 sy add def

/lw  0.6 mm def      % standard linewidth

%—Procedures

/Grid                % construction grid
{ newpath
  1 0 0 0 setcmykcolor
  0.1 mm setlinewidth
 /gx 0 def  /gdx 10 mm def
 /gy 0 def  /gdy 10 mm def
 { gx gy moveto
   gx bx add gy lineto stroke
  /gy gy gdy add def gy by 0.01 add gt {exit}if } loop
 /gy 0 def
 { gx gy moveto
   gx gy by add lineto stroke
  /gx gx gdx add def gx bx 0.01 add gt {exit}if } loop
} def

/Box
{ newpath
  0.95 setgray
  0  0 moveto
  bx  0 lineto
  bx by lineto
  0  by lineto
  closepath
  fill
} def


/FuncS
{/x kx x mul def          % mult x by kx,  store x
 /y x 360 mul sin def     % mult x by 360, store y=sin(x)
 /y ky y mul def          % mult y by ky,  store y
} def

/FuncC
{/x kx x mul def
 /y x 360 mul cos def
 /y ky y mul def
} def

%—Program

false setstrokeadjust

gsave
```

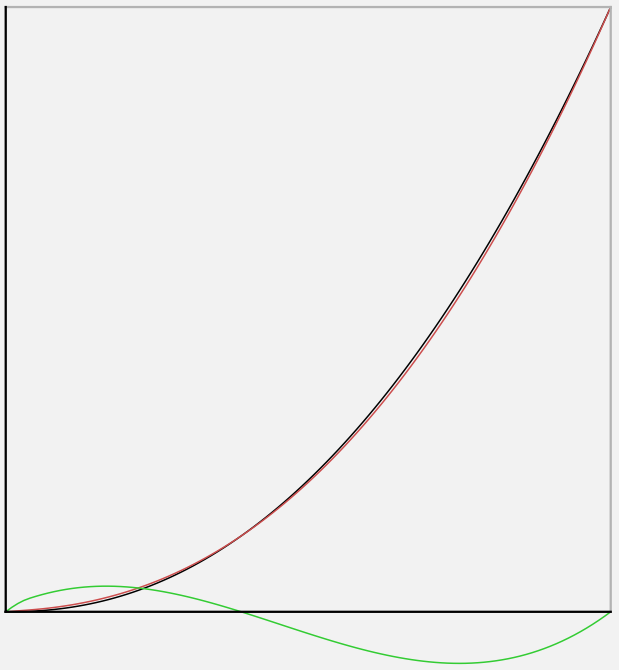


2.2 Sine-Cosine

```
Box
Grid

2 setlinecap  % 0=short 1=round 2=long

%—Frame
newpath
x0 ya moveto
xe ya lineto
x0 ye moveto
xe ye lineto
xe ya moveto
xe ye lineto
lw setlinewidth
0.7 setgray
stroke

%—global transformation

x0 y0 translate
sx sx scale
lw sx div setlinewidth
0 setlinecap

/x 0 def
FuncS x y moveto
0  1 n { pop
     FuncS
     x y lineto
     /x dx x add def } for
1 0 0 setrgbcolor
stroke

/x 0 def
FuncC x y moveto
0  1 n { pop
     FuncC x y lineto /x dx x add def } for
     0 0 1 setrgbcolor
[6 sx div 2 sx div] 0 setdash
stroke

grestore

2 setlinecap

%—Axes
newpath
x0 y0 moveto
xe y0 lineto
x0 ya moveto
x0 ye lineto
lw setlinewidth
0.0 setgray
stroke

showpage
```

3.1 Gamma Curves (Power functions)

```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox:  0 0 284 284
%%Creator:      Gernot Hoffmann
%%Title:        sRGB Curves
%%CreationDate: Dec.29,2002

%—Variables

/mm {2.834646 mul} def

/n  100 def
/dx   1 n  div def

/sx  80 mm def
/sy  80 mm def
/bx 100 mm def
/by 100 mm def


/x0  10 mm def
/y0  10 mm def

/xa  x0 def
/xe  x0 sx add def

/ya  y0 def
/ye  y0 sy add def

/x   0 def /y   0 def
/y1  0 def
/gx  0 def /gy  0 def
/gdx 0 def /gdy 0 def

%—Procedures

/Grid           % construction grid
{ newpath
  1 0 0 0 setcmykcolor
  0.1 mm setlinewidth
 /gx 0 def  /gdx 10 mm def
 /gy 0 def  /gdy 10 mm def
 { gx gy moveto
   gx bx add gy lineto stroke
  /gy gy gdy add def gy by 0.01 add gt {exit}if } loop
 /gy 0 def
 { gx gy moveto
   gx gy by add lineto stroke
  /gx gx gdx add def gx bx 0.01 add gt {exit}if } loop
} def

/Box
{ newpath
  0.95 setgray
  0    0 moveto
  bx   0 lineto
  bx  by lineto
  0   by lineto
  closepath
  fill
} def

/FuncG
{/y x 2.2 exp def
} def

/FuncS
{x 0.03928 le
 { /y x 12.92 div def }
 {/y x 0.055 add 1.055 div 2.4 exp def }ifelse
 } def

/FuncD
{ FuncG
  /y1 y def
  FuncS
  /y y y1 sub 10 mul def
} def
```

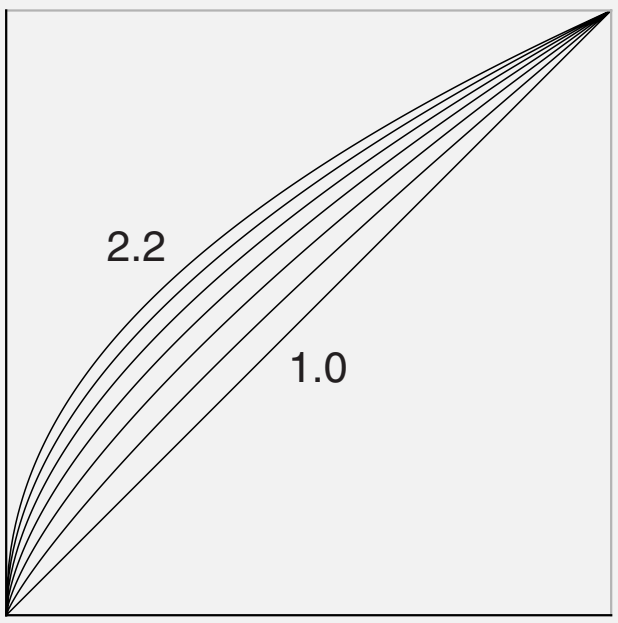


Gamma curves transform values C=R,G,B into new values C by a power function (1) or by a linear slope plus a power function (2)

(1) Black $C=C^{2.2}$

(2) Red sRGB

(3) Green 10 times the difference



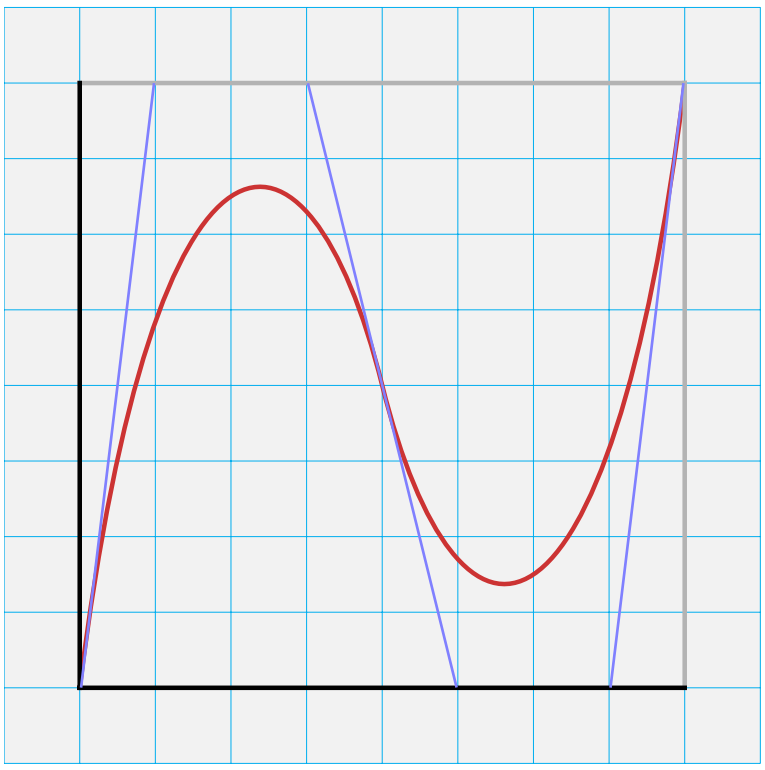
Inverse Gamma curves for G=1.. 2.2
$C = C^{1/G}$

3.2 Gamma Curves

```
%—Begin

gsave

true setstrokeadjust

Box
%Grid

false setstrokeadjust

x0 y0 translate
sx sx scale

0 setgray
0.2 mm sx div setlinewidth
/x 0 def
FuncG x y moveto
0 1 n { pop
        FuncG
        x y lineto
       /x dx x add def} for
stroke

0.8 0.3 0.3 setrgbcolor
0.2 mm sx div setlinewidth
/x 0 def
FuncS x y moveto
0 1 n { pop
        FuncS
        x y lineto
       /x dx x add def} for
stroke

0.2 0.8 0.2 setrgbcolor
0.2 mm sx div setlinewidth
/x 0 def
FuncD x y moveto
0 1 n { pop
        FuncD
        x y lineto
       /x dx x add def} for
stroke

grestore

2 setlinecap

%—frame
newpath
x0 ye moveto
xe ye lineto
xe ya moveto
xe ye lineto
0.3 mm setlinewidth
0.7 setgray
stroke

%—axes
newpath
x0 y0 moveto
xe y0 lineto
x0 ya moveto
x0 ye lineto
0.3 mm setlinewidth
0.0 setgray
stroke

showpage
```

4.1 Bézier Techniques

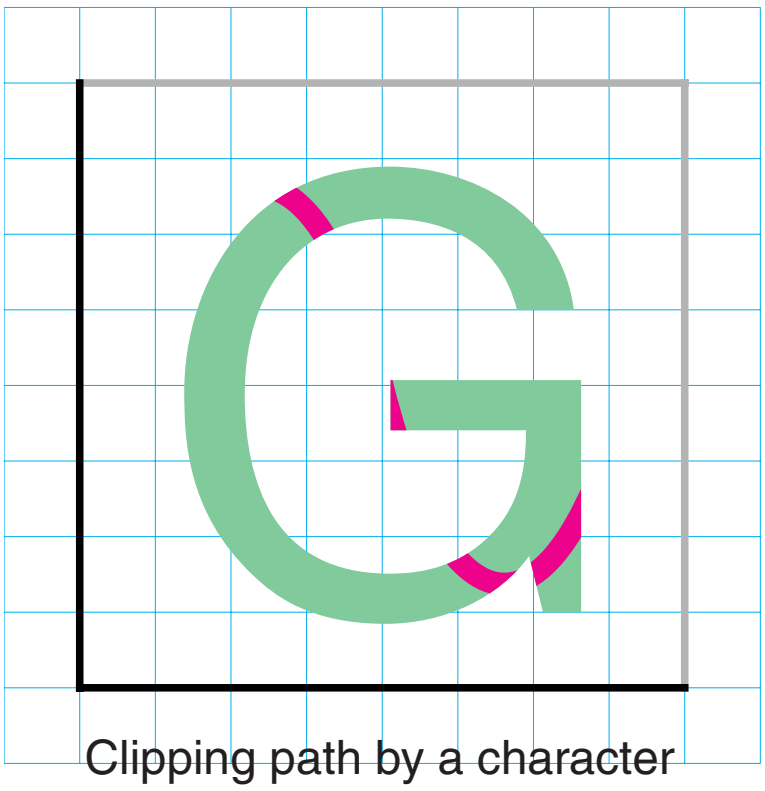
```
%!PS-Adobe-3.0       EPSF-3.0
%%BoundingBox:       0 0 284 284
%%Creator:           Gernot Hoffmann
%%Title:             Bezier Curve
%%CreationDate:      Nov.29,2002

%—Variables

/mm {2.834646 mul} def % points per mm

/sx  80 mm def
/sy  80 mm def
/bx 100 mm def
/by 100 mm def


/x0  10 mm def
/y0  10 mm def

/xa  x0 def
/xe  x0 sx add def

/ya  y0 def
/ye  y0 sy add def

/lw  0.6 mm def

/x1 0 def /y1 0 def
/x2 0 def /y2 0 def
/x3 0 def /y3 0 def

%—Procedures

/Grid     % construction grid
{ newpath
  1 0 0 0 setcmykcolor
  0.1 mm setlinewidth
 /gx 0 def  /gdx 10 mm def
 /gy 0 def  /gdy 10 mm def
 { gx gy moveto
   gx bx add gy lineto stroke
  /gy gy gdy add def gy by 0.01 add gt {exit}if }
loop
 /gy 0 def
 { gx gy moveto
   gx gy by add lineto stroke
  /gx gx gdx add def gx bx 0.01 add gt {exit}if }
loop
} def

/Box
{ newpath
  0.95 setgray
  0  0 moveto
  bx 0 lineto
  bx  by lineto
  0  by lineto
  closepath
  fill
} def

%—Begin

gsave

Box
Grid

x0 y0 translate
sx sx scale
lw sx div setlinewidth
```



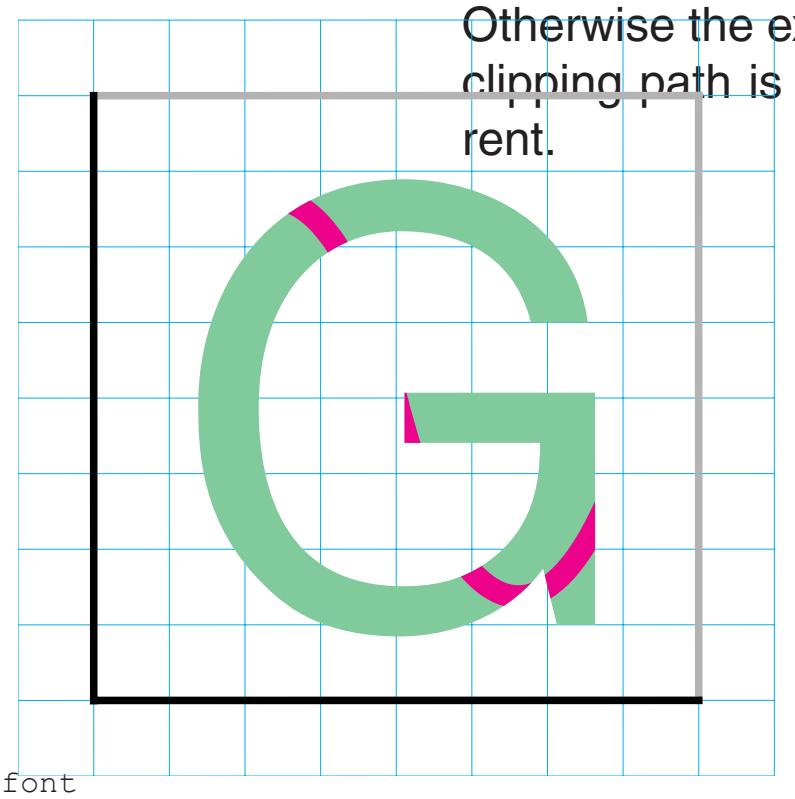
Control point and tangent drawing not by PostScript

5.1 Clipping Path

```
% Bezier    continued
 newpath
 0 0 moveto
/x1 0.125 def /y1 1.00 def
/x2 0.375 def /y2 1.00 def
/x3 0.500 def /y3 0.50 def
 x1 y1 x2 y2 x3 y3 curveto
/x1 0.625 def /y1 0.00 def
/x2 0.875 def /y2 0.00 def
/x3 1.000 def /y3 1.00 def
 x1 y1 x2 y2 x3 y3 curveto
0.8 0.2 0.2 setrgbcolor
stroke

grestore

2 setlinecap

%—frame
x0 ye moveto
xe ye lineto
xe ya moveto
xe ye lineto
lw setlinewidth
0.7 setgray
stroke

%—axes
x0 y0 moveto
xe y0 lineto
x0 ya moveto
x0 ye lineto
lw setlinewidth
0.0 setgray
stroke

showpage
```



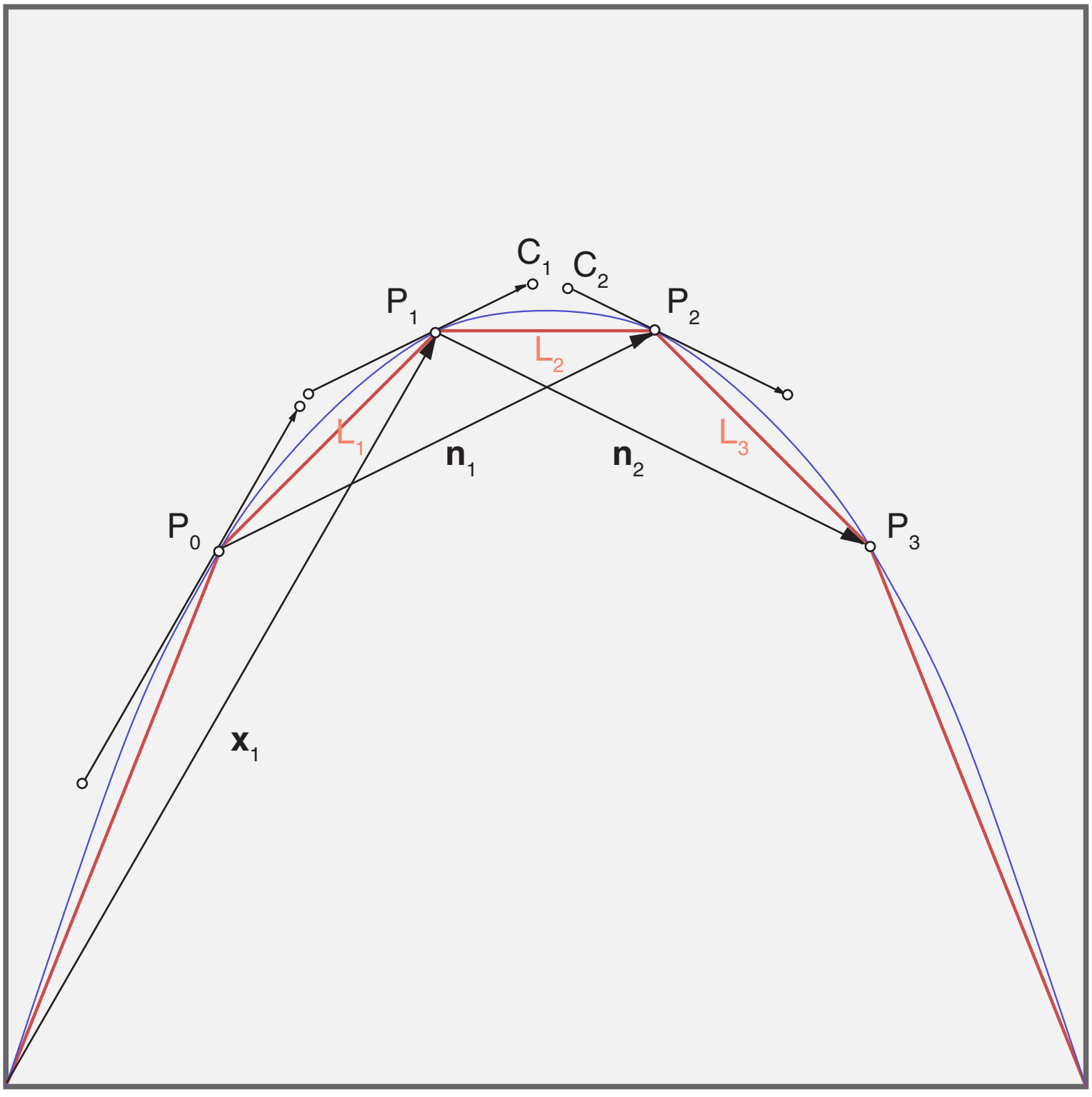
Clipping path by a character Test for transparency in PM. PageMaker requires a rotation of 0.01° or a mirroring Otherwise the exterior of the clipping path is not transparent.



```
%!PS-Adobe-3.0        EPSF-3.0
%%BoundingBox:        0,0,284,284
%%Creator:            Gernot Hoffmann
%%Title:              Clipping Path
%%CreationDate:       Dec.21,2002

%—Variables
% ...


%—clipping path
/Helvetica findfont 80 mm scalefont setfont
newpath
20 mm 20 mm moveto
(G) true charpath clip

Box

x0 y0 translate
sx sx scale

% ...
```

6.1 Polyline Bézier Interpolation

The task: replace a polyline (red) by a smooth function (blue), using PostScript Bézier interpolation.



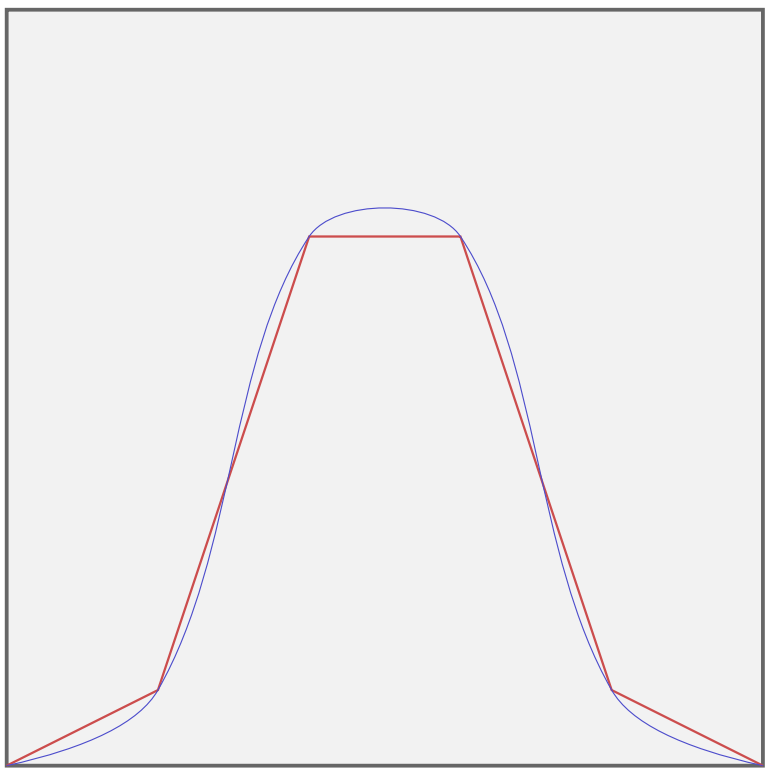
Maxim Shemanarev [6] had described a very simple method for finding reasonable control points for the Bézier construction. The result is not as good as a spline interpolation but much easier to program.

We consider a sequence of points P_0, P_1, P_2, P_3, represented by vectors $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}_2$ $\mathbf{x}_3$.
P_0 is not the first point of the curve, it´s the first point of the actual sequence.

1. Connect P_0 and P_2 by $\mathbf{n}_1$
2. Move $\mathbf{n}_1$ to P_1 and multiply by a factor, mostly K = 0.3 ... 0.4
3. Shift the center of $\mathbf{n}_1$ along the tangent direction according to the proportion of the lengths L_1 and L_2. This delivers the control point C_1.
4. Construct the control point C_2 similarly.
5. Draw from actual point P_1 to P_2 by PostScript *curveto .*

6.2 Polyline Bézier Interpolation

```
%!PS-Adobe-3.0       EPSF-3.0
%%BoundingBox:       0 0 341 341
%%Creator:           Gernot Hoffmann
%%Title:             Bezier Curve
%%CreationDate:      Dec.08,2002

% Bezier with endsegment reconstruction

/mm { 2.834646 mul } def

/ci [ 0 0.0 0.0
      1 0.2 0.1
      2 0.4 0.7
      3 0.6 0.7
      4 0.8 0.1
      5 1.0 0.0   ] def

/sx 100 mm def
/sy 100 mm def
/bx 120 mm def
/by 120 mm def

/x0  10 mm def
/y0  10 mm def
```



```
/Bezier
{ % Table ci: parameter,x,y
  % Reconstruction of two control points at the end segments
  % from one available control point at each end
  newpath
  0.3 0.3 0.8 setrgbcolor
  0.15 mm sx div setlinewidth
 /kf 0.35 def  % 0.3 ..0.5; optimized for this example
 /k 0 def
 /x0 ci 1 get def
 /y0 ci 2 get def
 /x1 ci 4 get def
 /y1 ci 5 get def
 /x2 ci 7 get def
 /y2 ci 8 get def
  x0 y0 moveto
 /n1x x2 x0 sub kf mul def
 /n1y y2 y0 sub kf mul def
 /L1  x1 x0 sub dup mul y1 y0 sub dup mul add sqrt def
 /L2  x2 x1 sub dup mul y2 y1 sub dup mul add sqrt def
 /c2x x1 L1 L1 L2 add div n1x mul sub def
 /c2y y1 L1 L1 L2 add div n1y mul sub def
 /c1x x0 c2x 2 mul add 3 div def % Reconstruction
 /c1y y0 c2y 2 mul add 3 div def
 /c2x x1 c2x 2 mul add 3 div def
 /c2y y1 c2y 2 mul add 3 div def
  c1x c1y c2x c2y x1 y1 curveto
  1 1 ci length 3 idiv 3 sub
 {  pop
    /x3  ci k 10 add get def
    /y3  ci k 11 add get def
    /n2x x3 x1 sub kf mul def
    /n2y y3 y1 sub kf mul def
    /L3  x3 x2 sub dup mul y3 y2 sub dup mul add sqrt def
    /c1x x1 L2 L1 L2 add div n1x mul add def
    /c1y y1 L2 L1 L2 add div n1y mul add def
    /c2x x2 L2 L3 L2 add div n2x mul sub def
    /c2y y2 L2 L3 L2 add div n2y mul sub def
     c1x c1y c2x c2y x2 y2 curveto
    /k k 3 add def
    /x1  x2  def
    /y1  y2  def
    /x2  x3  def
    /y2  y3  def
    /n1x n2x def
    /n1y n2y def
    /L1  L2  def
    /L2  L3  def
  } for
/c1x x1 L2 L1 L2 add div n1x mul add def
/c1y y1 L2 L1 L2 add div n1y mul add def
/c2x x2 c1x 2 mul add 3 div def % Reconstruction
/c2y y2 c1y 2 mul add 3 div def
/c1x x1 c1x 2 mul add 3 div def
/c1y y1 c1y 2 mul add 3 div def
 c1x c1y c2x c2y x2 y2 curveto
 stroke
} def
```

6.3 Polyline Bézier Interpolation

```
/Box
{ newpath
  0.95 setgray
  x0 y0 moveto sx 0 rlineto 0 sy rlineto sx neg 0 rlineto
  closepath gsave fill grestore
  0.4 setgray 0.5 mm setlinewidth stroke
} def

/Polyline
{ newpath
  0.8 0.3 0.3 setrgbcolor
  0.15 mm sx div setlinewidth
 /k 0 def
 /x0 ci 1 get def
 /y0 ci 2 get def
  x0 y0 moveto
  1 1 ci length 3 div 1 sub
  { /k  k 3 add def
    /x1 ci k 1 add get def
    /y1 ci k 2 add get def
     x1 y1 lineto
  } for
  stroke
} def

gsave

Box

x0 y0 translate
sx sx scale

Polyline
Bezier

grestore

showpage
```

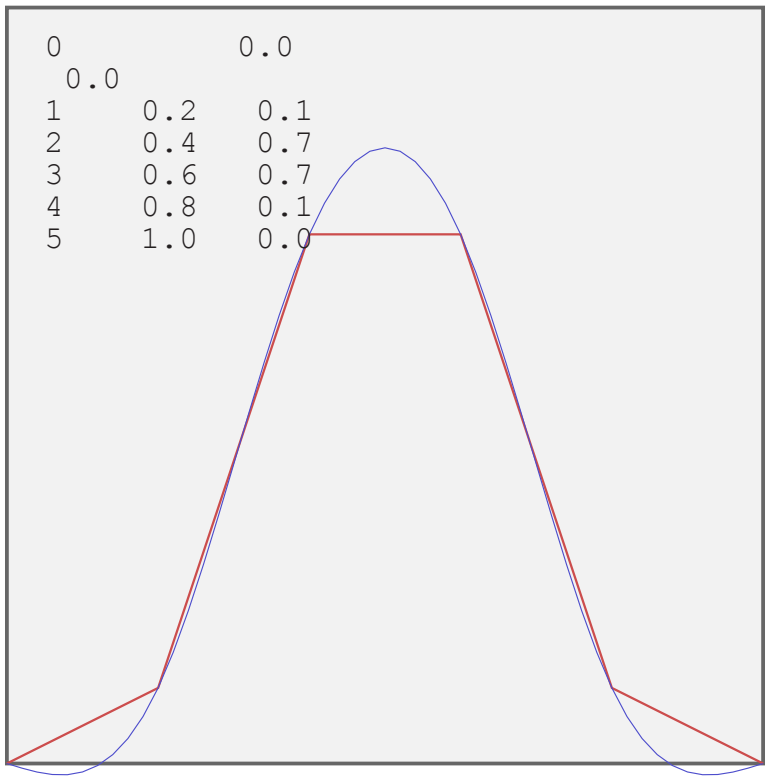
7.1 Polyline Spline Interpolation

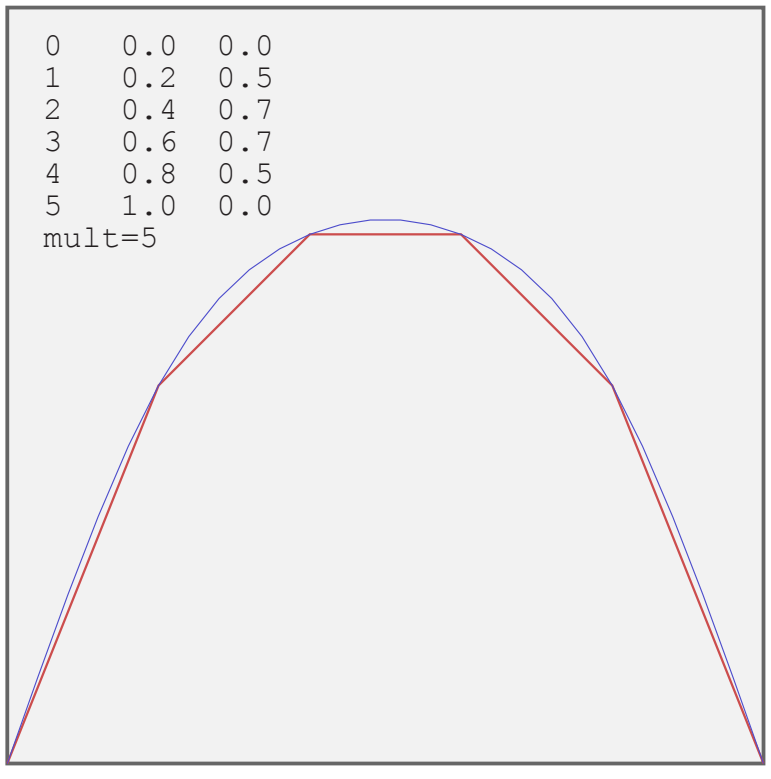
By 'spline' we mean here a global spline, opposed to local Bézier splines. These suffer from too many degrees of freedom, whereas a global spline is optimized for least curvature.

The PostScript source code is based on a Pascal program [7] and the theory in [8].

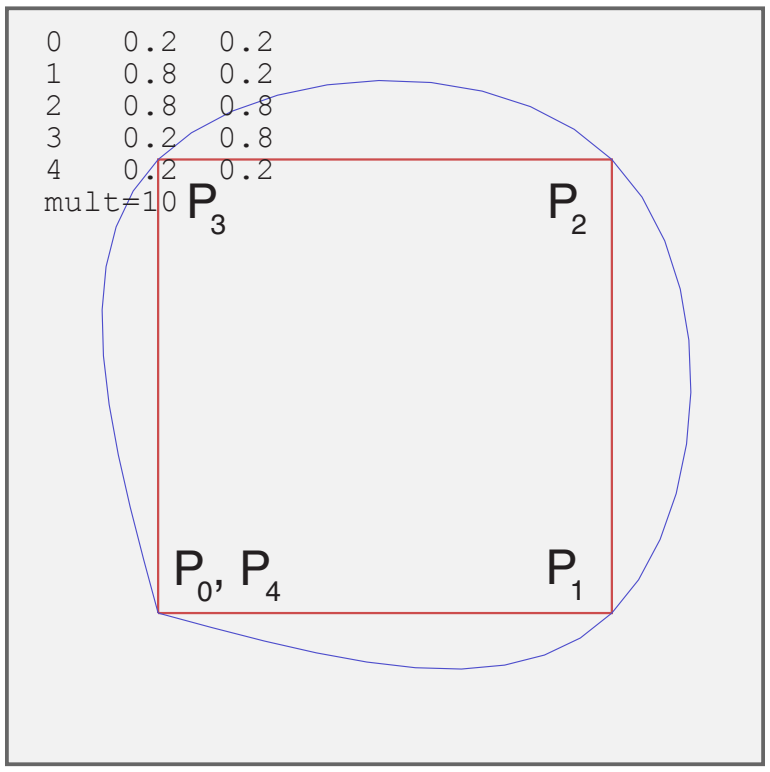
The spline interpolator inserts new points between the polyline points.
The parameter *mult=5* means, that 4 new points are inserted between two old points.

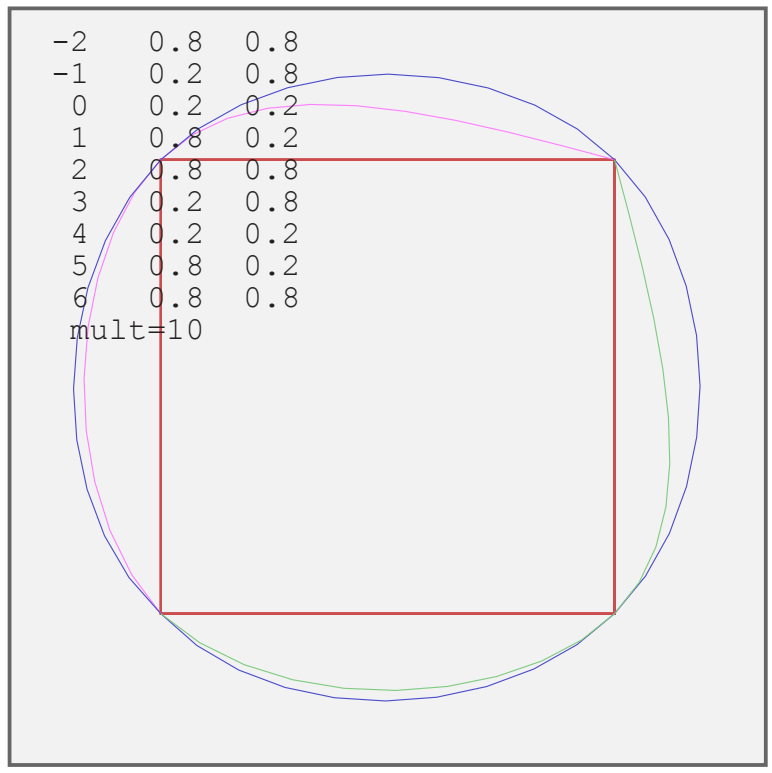
The first example (same as used for Bézier) shows an unpleasant overshoot at both ends.





The next example shows the interpolation of four points. Because the ends of the spline have zero curvature, the whole contour does not look 'round' (left image).
We can add the first points P_1, P_2 to the list as endpoints P_5, P_6 and insert the last points P_2, P_3 as new first points P_{-1}, P_{-2}. Then we achieve a perfectly balanced cyclical spline.





7.2 Polyline Spline Interpolation

```
%!PS-Adobe-3.0      EPSF-3.0
%%BoundingBox:      0 0 341 341
%%Creator:          Gernot Hoffmann
%%Title:            Spline Curve
%%CreationDate:     Dec.08,2002

% Given a table /fi of 0..fp rows
% Each row:      parameter, x, y
% Create for pxy=1 table vecx with 0..spl interpolated values
% Create for pxy=2 table vecy with 0..spl interpolated values
% spl is practically limited, e.g. 1000
% mult=5 means: insert 4 new values between two old values
% mult=1,2 ...
% E.g. mult=5
%      fp  =5
% Raw data in fi column 2 or 3    0    1    2    3    4    5
% Interpolation in vecx or vecy   012345678901234567890123 45

%—Variables

/fi [  0   0.0  0.0
       1   0.2  0.5
       2   0.4  0.7
       3   0.6  0.7
       4   0.8  0.5
       5   1.0  0.0   ] def

/mult 5 def
/fp   fi length 3 idiv 1 sub def
/spl  fp mult mul def
/spa  spl 1 add def

% e.g. array size spa=26
/vecx spa array def
/vecy spa array def

/mm { 2.834646 mul } def

/sx 100 mm def
/sy 100 mm def
/bx 120 mm def
/by 120 mm def

/x0  10 mm def
/y0  10 mm def

%—Procedures

/Box
{ newpath
  0.95 setgray
  x0  y0    moveto
  sx   0   rlineto
  0   sy   rlineto
  sx neg 0 rlineto
  closepath
  gsave
  fill
  grestore
  0.4 setgray
  0.5 mm setlinewidth
  stroke
} def

/Polyline
{ newpath
  0.8 0.3 0.3 setrgbcolor
  0.15 mm sx div setlinewidth
 /k 0 def
 /x0 fi 1 get def
 /y0 fi 2 get def
  x0 y0 moveto
  1 1 fp
  { /k  k 3 add def
    /x1 fi k 1 add get def
    /y1 fi k 2 add get def
     x1 y1 lineto
  } for
  stroke
} def
/MakeSpline
```

7.3 Polyline Spline Interpolation

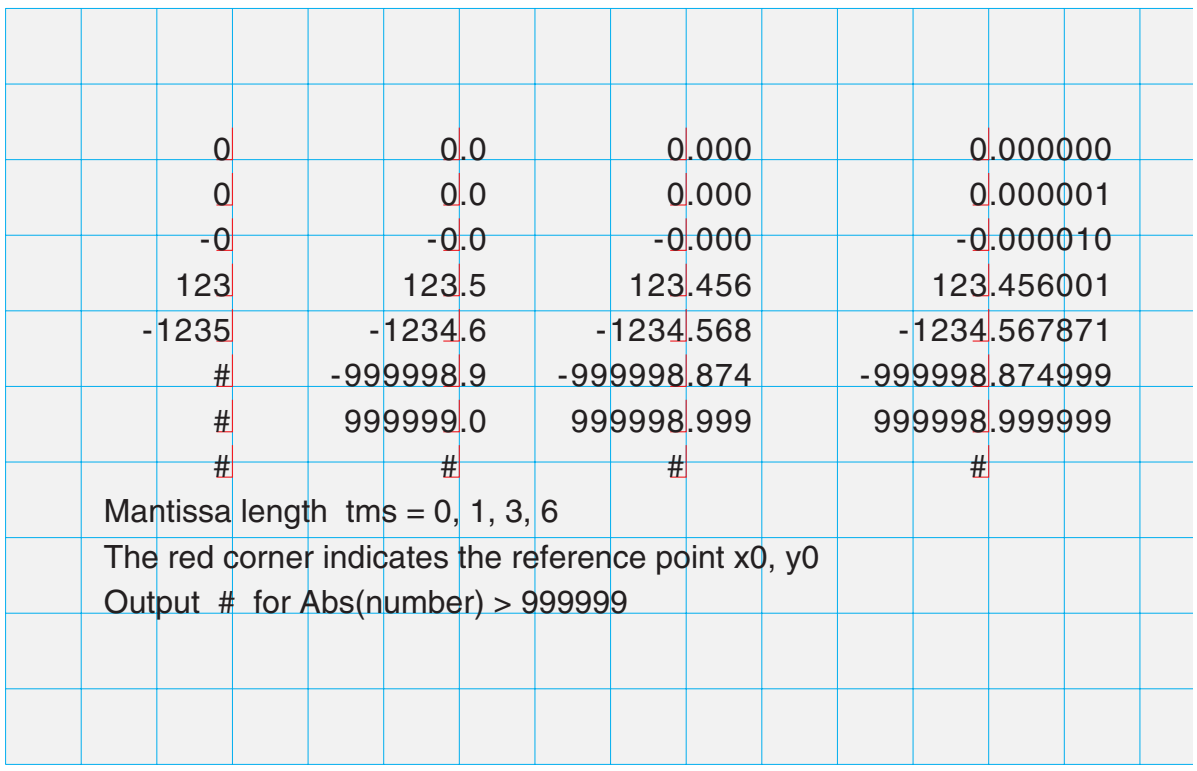
```
{ % table fi: par,x,y
% Arrays
  /fu  spa array def
  /mi  spa array def
  /re  spa array def
  /yy  spa array def
  /xx  spa array def
% Constants
  /n   fp 1 sub def
  /hh  mult def
  /hd6 hh 6 div def
  /hm6 hh 6 mul def
  /h6h -6 hh dup mul div def
% Copy x or y values with gaps
  pxy 1 eq
  { 0 1 fp
    {/k exch def % loop counter
      fu k hh mul fi k 3 mul 1 add get put } for
  } if
  pxy 2 eq
  { 0 1 fp
    {/k exch def
      fu k hh mul fi k 3 mul 2 add get put } for
  } if
% Spline
   mi 1 0.25 put
   1 1 n 1 sub
   { /j exch def
     mi j 1 add 1 4 mi j get sub div put   } for
   1 1 n
   { /j exch def
    /jh j hh mul def
     re j fu jh hh add get fu jh get 2 mul sub fu jh hh sub get add h6h mul put } for
   yy 1 re 1 get put
   2 1 n
   { /j exch def
     yy j re j get yy j 1 sub get mi j 1 sub get mul sub put } for
% Tridiagonal equation
   xx 0 0 put
   xx n yy n get mi n get mul neg put
   xx n 1 add 0 put
   n 1 sub -1 1
  { /j exch def
    xx j yy j get xx j 1 add get add mi j get mul neg put } for
  /i 0 def
   0 1 n
  { /j exch def % loop counter
    /jh j hh mul def
    /a3 xx j 1 add get xx j get sub hm6 div def
    /a2 xx j get 0.5 mul def
    /a1 fu jh hh add get fu jh get sub hh div xx j 1 add get xx j get 2 mul add hd6 mul sub def
    /a0 fu jh get def
     1 1 hh 1 sub
     { /k exch def
       /i i 1 add def
         fu i a3 k mul a2 add k mul a1 add k mul a0 add put } for
    /i i 1 add def
  } for
% Copy
 pxy 1 eq
 { fu vecx copy } if
 pxy 2 eq
 { fu vecy copy } if
} def


/DrawSpline
{ newpath
  0.3 0.3 0.8 setrgbcolor
  0.15 mm sx div setlinewidth
  vecx 0 get vecy 0 get moveto
 /k 1 def
  1 1 spl
  { pop
    vecx k get vecy k get lineto
   /k k 1 add def
  } for
  stroke
} def
%—Begin
```

7.4 Polyline Spline Interpolation

```
gsave

Box

x0 y0 translate
sx sx scale

Polyline

/pxy 1 def MakeSpline
/pxy 2 def MakeSpline

DrawSpline

grestore

showpage
```

8.1 Drawing Numbers

```
%!PS-Adobe-3.0   EPSF-3.0
%%BoundingBox:   0 0 445 284
%%Creator:       Gernot Hoffmann
%%Title:         Numbers2
%%CreationDate:  April 29 / 2005

%—Variables

/mm {2.834646 mul} def % points per mm

/bx 160 mm def
/by 100 mm def

/nums [         0.0
                0.000001
               -0.00001
              123.456
            -1234.567890
          -999998.90
           999999.00
           999999.10      ] def

%—Procedures
```



Drawing numbers equally spaced and aligned
by Helvetica for diagrams

```
/Grid           % construction grid
{ newpath
  1 0 0 0 setcmykcolor
  0.1 mm setlinewidth
 /gx 0 def  /gdx 10 mm def
 /gy 0 def  /gdy 10 mm def
 { gx gy moveto
   gx bx add gy lineto stroke /gy gy gdy add def gy by 0.01 add gt {exit}if } loop
 /gy 0 def
 { gx gy moveto
   gx gy by add lineto stroke /gx gx gdx add def gx bx 0.01 add gt {exit}if } loop
} def

/Box
{ newpath
  0.95 setgray
  0    0 moveto
  bx   0 lineto
  bx  by lineto
  0   by lineto
  closepath
  fill
} def

/Shownum
% Draw number by string
% Version May 15 2004 / uses rounding
%
%  tx0 ty0 nu :on stack
%
%  tx0    Position                                    on stack
%  ty0                                                on stack
%  nu     Input number                 nu =+-999999   on stack
%  fh     Font height                                 global, not overwritten
%  tms    Mantissa number of characters    tms=0...6  global, not overwritten
%  tms=3  Example
%  input  -23.56789  -999.99    0.4567  9999.123456
%  result -23.568    -999.990   0.467   9999.123
%  Postscript number to string is not well defined
%  e.g. 1E-5 instead of 0.00001
%  We use a straightforward BCD conversion.
%  This is always affected by round-off errors
%  because of 32-bit arithmetic
%  Results are different, depending on the interpreter

{ /nu  exch def
  /ty0 exch def
  /tx0 exch def
  /tfw fh 0.6 mul def                                  % character distance
  /tna nu abs 10 tms neg exp 0.500001 mul add def

  /tdec 1E5 def
  /tchr 1 string def

   tna 999999.1 lt                                     % larger number replaced by #
  /tmm true def                                        % sign
  {/tx0 tx0 tfw 6 mul sub def
   /tz 0 def
  1 1 5                                                % first 5 digits, no leading 0
```

8.2 Drawing Numbers

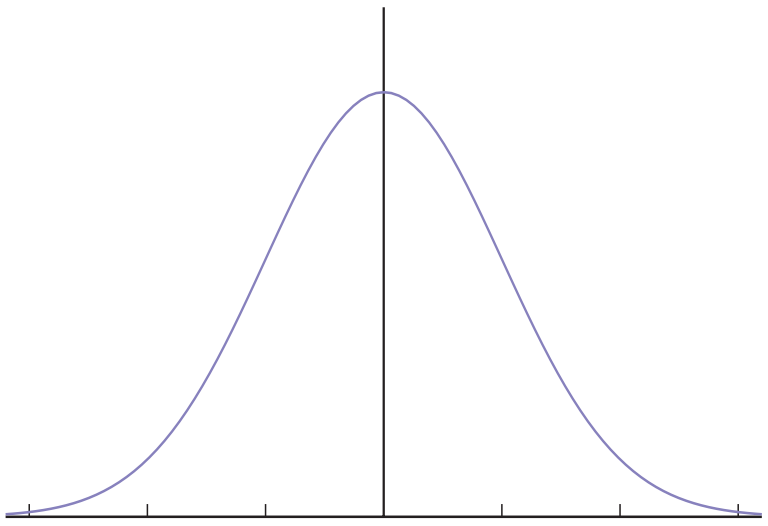
```
{ pop
 /tk 0 def
 { tna tdec gt {/tna tna tdec sub def /tk tk 1 add def}{exit} ifelse
 } loop
  tk 0 ne {/tz tz 1 add def} if
  tz 0 ne
  { tx0 ty0 moveto tk tchr cvs show
  } if
  tz 1 eq nu 0 lt and tmm and                          % minus  Correction 28/03/2005
  { tx0 tfw 0.7 mul sub ty0 moveto (-) show
   /tmm false def
  } if
 /tdec tdec 0.1 mul def
 /tx0 tx0 tfw add def
} for
 /tk 0 def                                             % leading 0
 { tna tdec gt {/tna tna tdec sub def /tk tk 1 add def}{exit} ifelse
 } loop
   tmm nu 0 lt and                                     % minus
  { tx0 tfw 0.7 mul sub ty0 moveto (-) show
  } if
    tx0 ty0 moveto tk tchr cvs show
   /tdec tdec 0.1 mul def
   /tx0 tx0 tfw add def
  tms 0 gt                                             % for float
  { tx0 ty0 moveto (.) show
   /tx0 tx0 tfw 0.5 mul add def
  1 1 tms
  { pop
   /tk 0 def
   { tna tdec gt {/tna tna tdec sub def /tk tk 1 add def}{exit} ifelse
   } loop
     tx0 ty0 moveto tk tchr cvs show
   /tdec tdec 0.1 mul def
   /tx0 tx0 tfw add def
  } for
  } if
 }{ tx0 tfw sub ty0 moveto (#) show} ifelse
} def

/RefEdge
{ x0 fh 0.5 mul sub y0 moveto x0 y0 lineto
  x0 y0 fh add lineto
  currentcmykcolor
  0 1 1 0 setcmykcolor
  stroke
  setcmykcolor} def

/Column
{ /k 0 def
   1 1 8
 { pop
  /nu nums k get def
   RefEdge
   x0 y0 nu Shownum
  /k   k  1 add def
  /y0 y0 dy sub def } for } def

%—Program

true setstrokeadjust

Box
Grid

false setstrokeadjust
/fh 12 def
```

8.3 Drawing Numbers

```
/Helvetica findfont fh scalefont setfont
 0 0 0 1 setcmykcolor
 0.15 mm setlinewidth
/x0  30 mm def
/y0  80 mm def
/dy   6 mm def
/tms  0 def      Column

/x0  60 mm def
/y0  80 mm def
/tms  1 def      Column

/x0  90 mm def
/y0  80 mm def
/tms  3 def      Column

/x0 130 mm def
/y0  80 mm def
/tms  6 def      Column

/x0  13 mm def
/y0  y0 dy def
x0 y0 moveto (Mantissa length  tms = 0, 1, 3, 6) show
/y0  y0 dy sub def
x0 y0 moveto (The red corner indicates the reference point x0, y0) show
/y0  y0 dy sub def
x0 y0 moveto (Output  #  for Abs(number) > 999999) show

showpage
```

A newer version in the Matrix Library [11] uses a dictionary for local variables. The algorithms are the same.

9.1 Gaussian Distribution

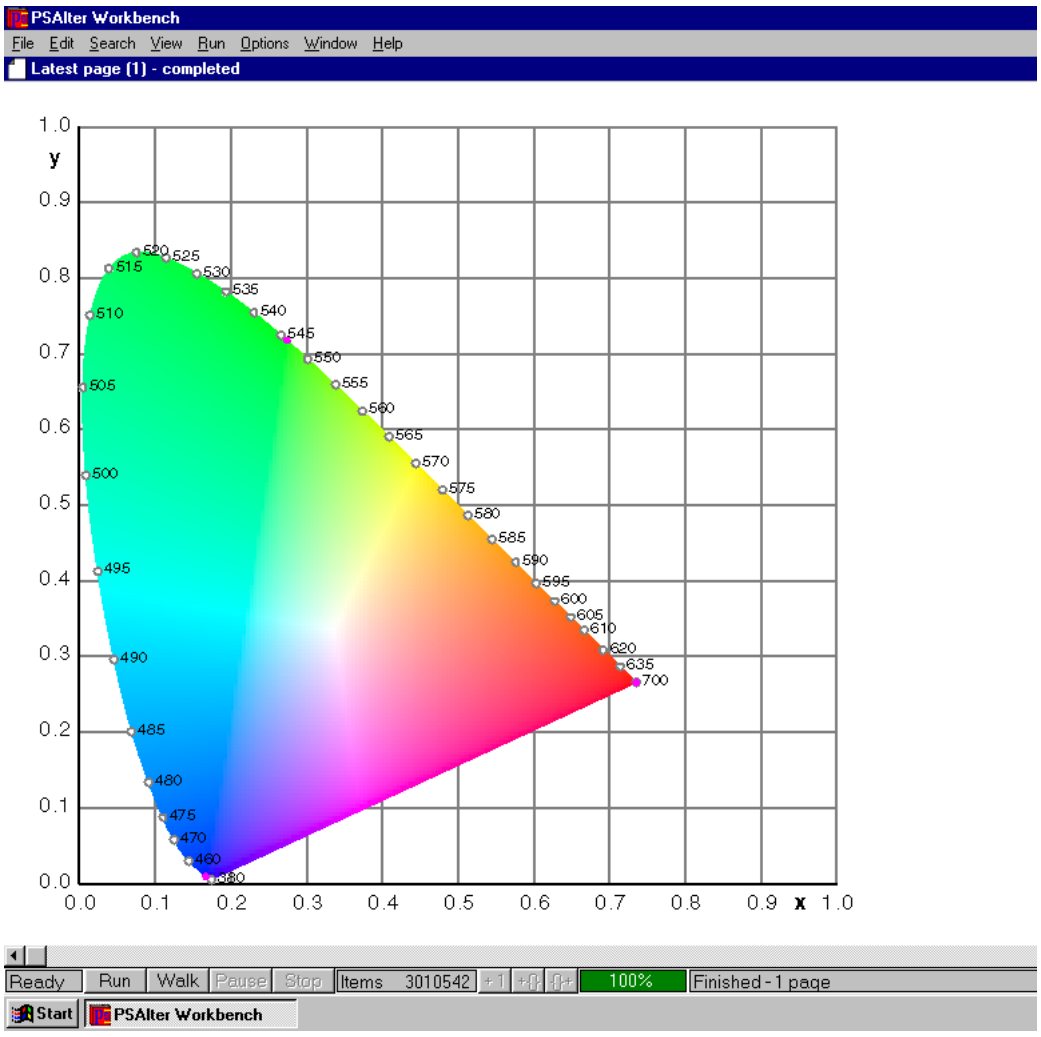


```
%!PS-Adobe-3.0      EPSF-3.0
%%BoundingBox:      0 0 340 227
%%Creator:          Gernot Hoffmann
%%Title:            Gauss Normal
%%CreationDate:     Nov.29,2002

/mm {2.834646 mul} def % points per mm

/n   50 def              % steps of x
/dx   1 n div  def

/x0  60 mm def           % center
/y0  10 mm def
/sx  50 mm def           % length of x-axis

/bx 120 mm def           % box
/by  80 mm def

/xa  x0 sx sub def       % initial
/xe  x0 sx add def       % end

/ya  10 mm def
/ye  70 mm def

/eul 2.718282 def        % e
/sig 1 3.2 div def  % sigma=1/3.2 of axis length

%—Procedures

/Grid                    % construction grid
{ newpath
  1 0 0 0 setcmykcolor
  0.1 mm setlinewidth
 /gx 0 def  /gdx 10 mm def
 /gy 0 def  /gdy 10 mm def
 { gx gy moveto
   gx bx add gy lineto stroke
  /gy gy gdy add def gy by gt {exit}if } loop
 /gy 0 def
 { gx gy moveto
   gx gy by add lineto stroke
  /gx gx gdx add def gx bx gt {exit}if } loop
} def

/Box
{ newpath
  0.95 setgray
  0    0 moveto
  bx   0 lineto
  bx  by lineto
  0   by lineto
  closepath
  fill
} def

%       y = e^( -0.5 Sqr(x/sig) )
/FuncG
{/y eul x sig div dup mul 0.5 mul neg exp def}def

gsave

% Box
% Grid

%-axes
newpath
0.3 mm setlinewidth
0 0 0 1 setcmykcolor
xa ya moveto xe ya lineto
x0 ya moveto x0 ye lineto
stroke
```

9.2 Gaussian Distribution

```
%tic marks
0.2 mm setlinewidth
/dt 1.5 mm def
/x x0 def
1 1 3
{/x x sx sig mul add def
  x y0 moveto 0 dt rlineto } for
/x x0 def
1 1 3
{/x x sx sig mul sub def
x y0 moveto 0 dt rlineto } for
stroke

x0 y0 translate                % local coordinates
sx sx scale

0.3 mm sx div setlinewidth
0.5 0.5 0 0 setcmykcolor

/x 0 def
FuncG x y moveto
1 1 n
{ pop /x x dx add def FuncG x y lineto } for
stroke

/x 0 def
FuncG x y moveto
1 1 n
{ pop /x x dx sub def FuncG x y lineto } for
stroke

showpage
```

10.1 CIE Chromaticity Diagrams



Very accurate CIE xyY Chromaticity Diagram
CIE primaries are shown by monitor primaries
Inverse Gamma corrected for Gamma=2.2
Contour interpolation

The EPS file for the complete diagram
can be loaded here:

http://www.fho-emden.de/~hoffmann/ciesuper.txt

Please rename then to
ciesuper.eps

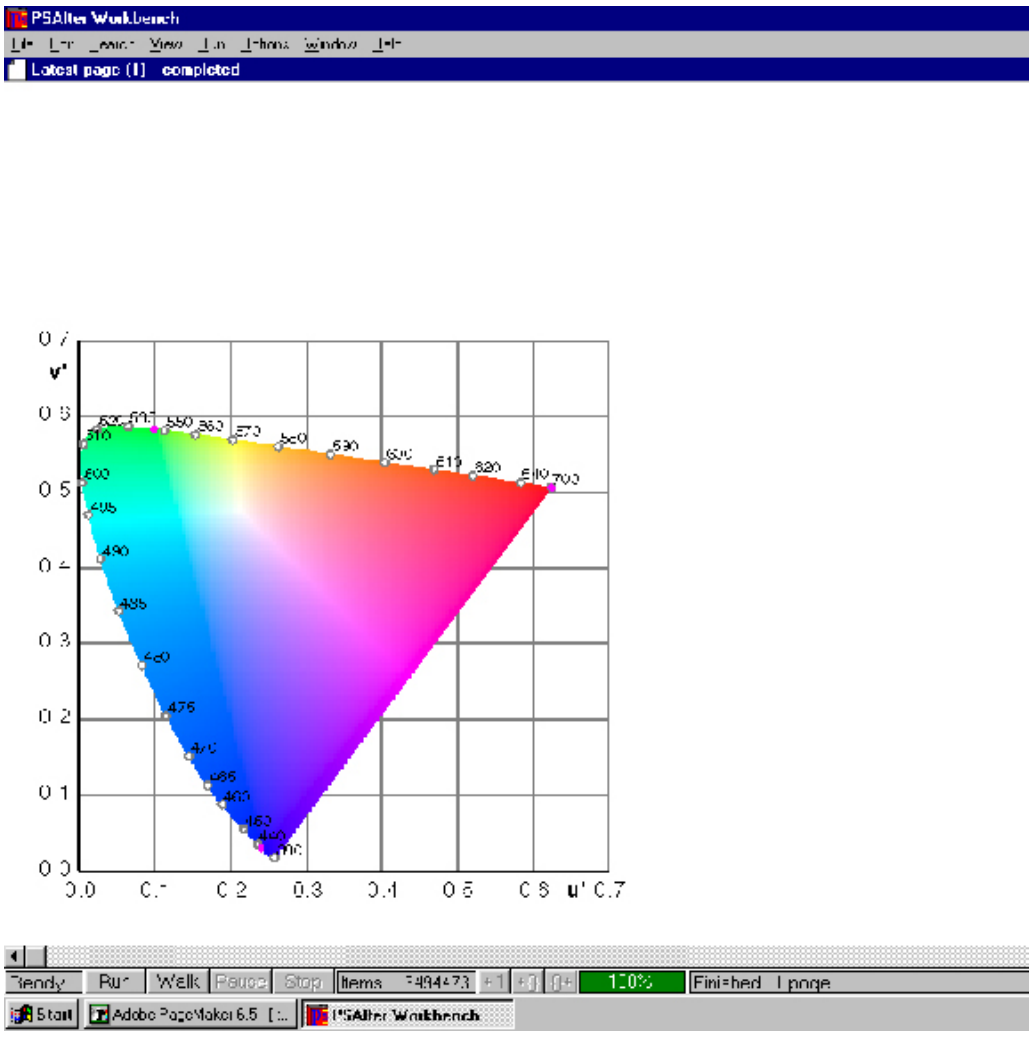
The EPS requires external fonts
Helvetica
Helvetica-Bold

The EPS does not have a TIFF preview

Printing may shift the white point
This depends on the assigned profile
CIE white point is at xw=1/3, yw=1/3

The image is a screenshot TIFF
Embedding the EPS would slow down Acrobat

10.2 CIE Chromaticity Diagrams



Very accurate CIE u' v' Chromaticity Diagram
CIE primaries are shown by monitor primaries
Inverse Gamma corrected for Gamma=2.2
Contour interpolation

The EPS file for the complete diagram
can be loaded here:

http://www.fho-emden.de/~hoffmann/cieuv.txt

Please rename then to
cieuv.eps

The EPS requires external fonts
Helvetica
Helvetica-Bold

The EPS does not have a TIFF preview

Printing may shift the white point
This depends on the assigned profile
CIE white point is at
uw=0.210526, vw=0.473684

The image is a screenshot
Embedding the EPS would slow down Acrobat

11.1 3D Graphics for Terrains

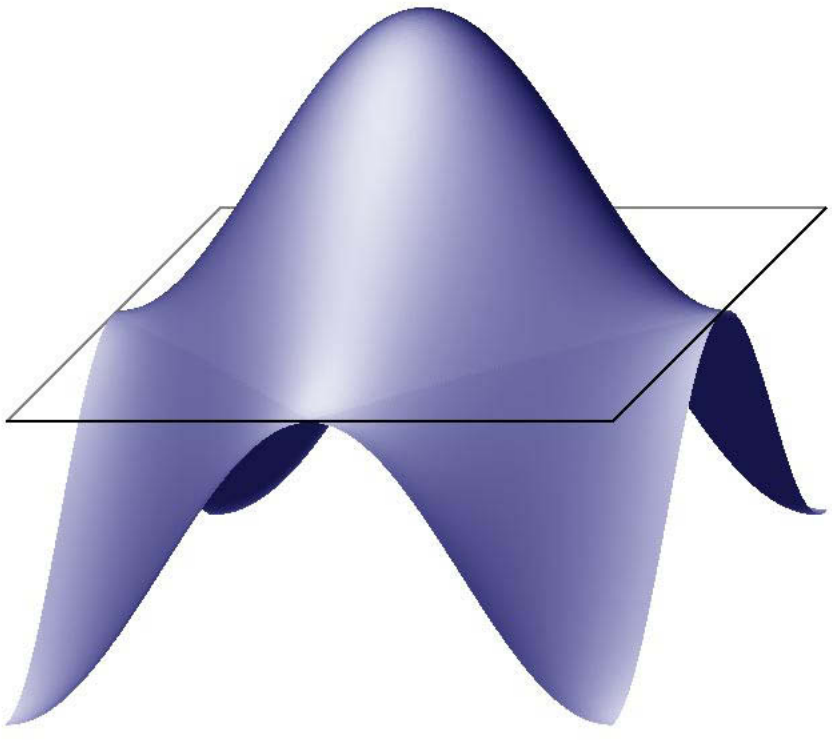
PostScript cannot draw 3D vector graphics directly. We encounter two problems:

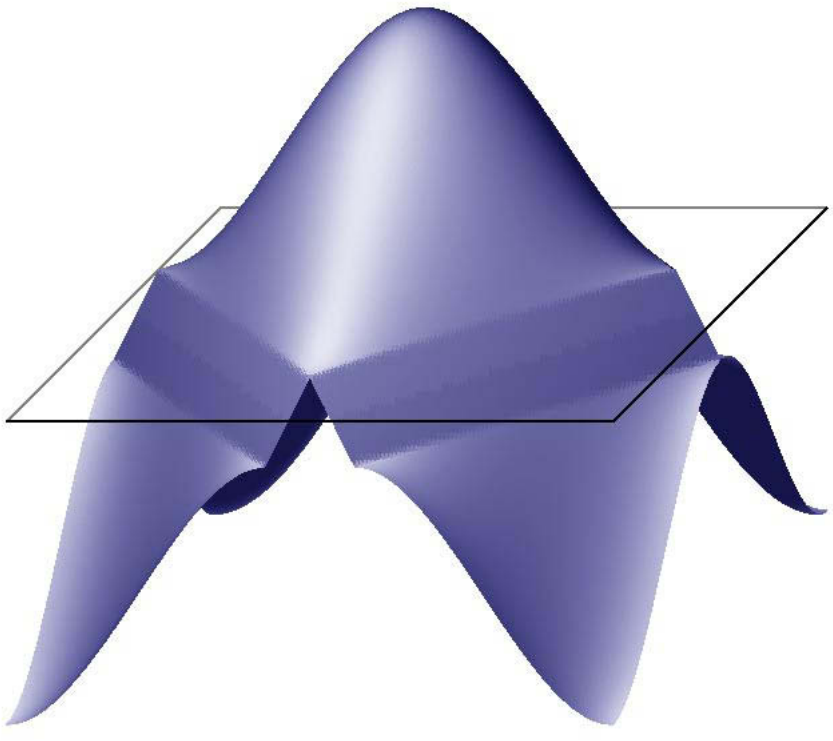
1. No hidden surface algorithm
2. No gradients in PS Level 2

We use here a cabinet projection and draw from back to front. Thus near parts hide remote parts. Small quadrilaterals are drawn by flat (façetted) shading instead of Gouraud shading. If the areas are small enough then the object looks smooth.

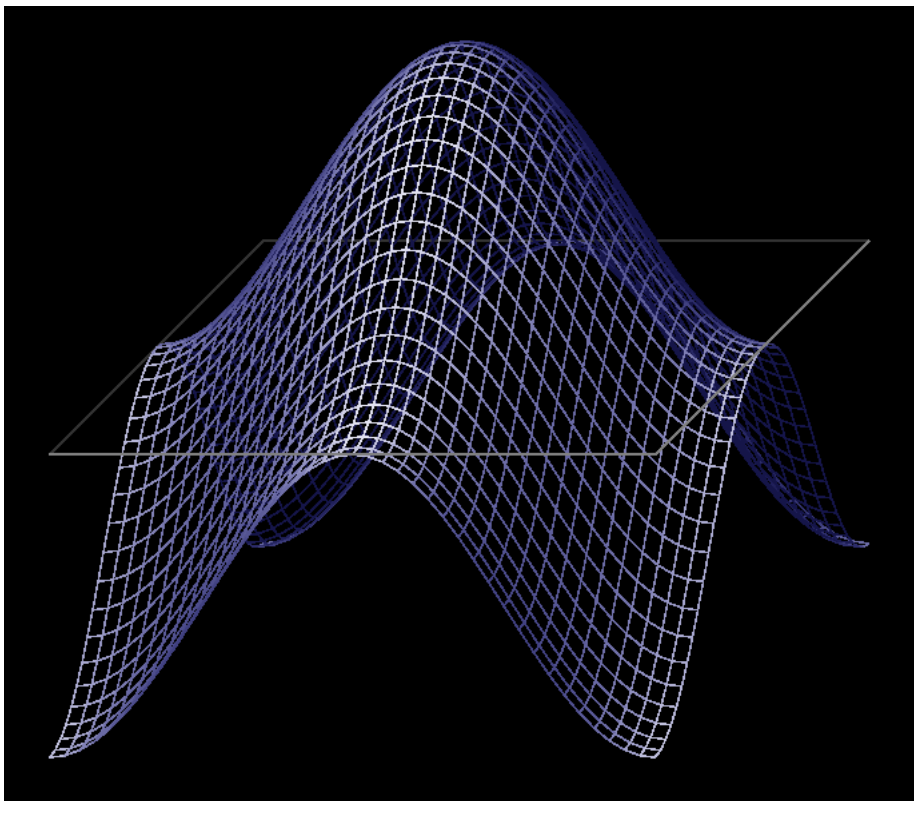
The examples show so-called Spot Functions. A Spot Function is used for halftoning. The interior of the cross section of the function z(x,y) and a horizontal plane delivers the spot area, e.g. for black. The left function is used for round spots, the right for elliptic spots.
Sources: round spot [1]. Elliptic spot PS code by Helge Blischke, Adobe Forum PS.
All images here are TIFFs. Embedded EPSs would slow down Acrobat considerably.

Best view for TIFFs: zoom 100% / 72 dpi





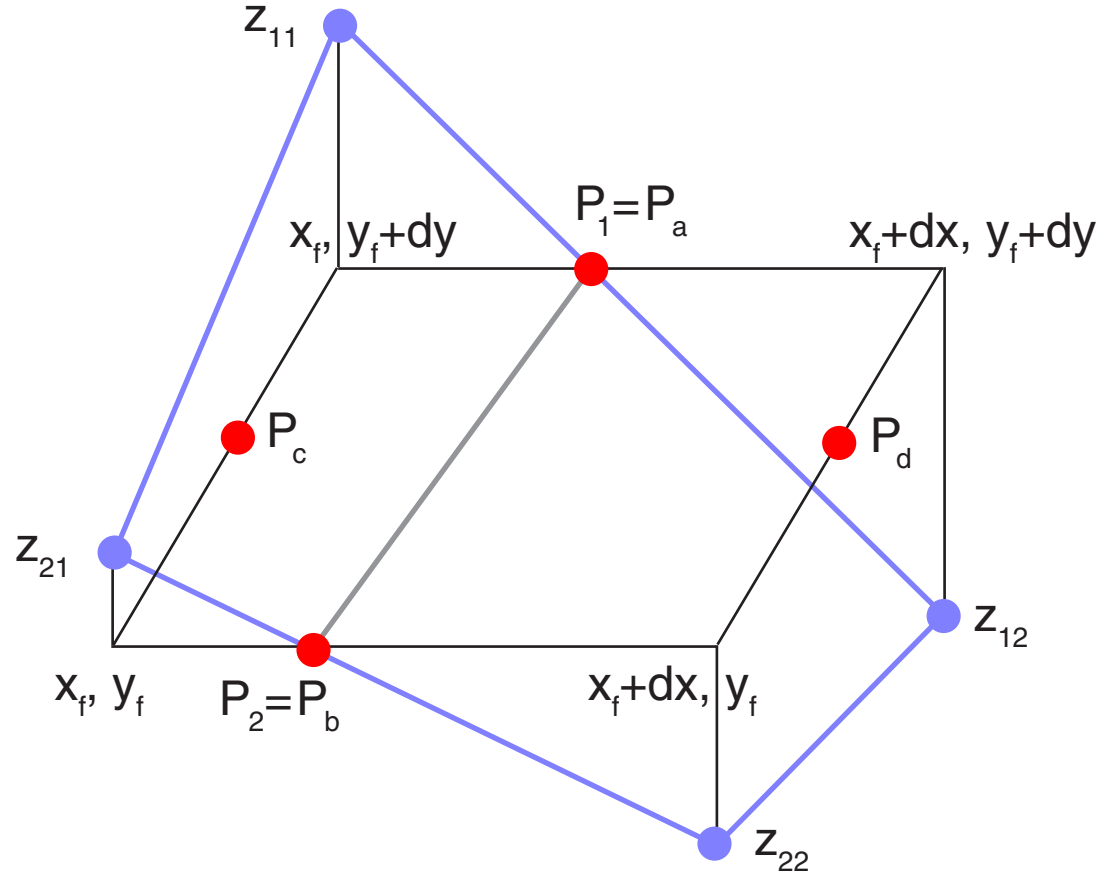
Round spots



Elliptic spots

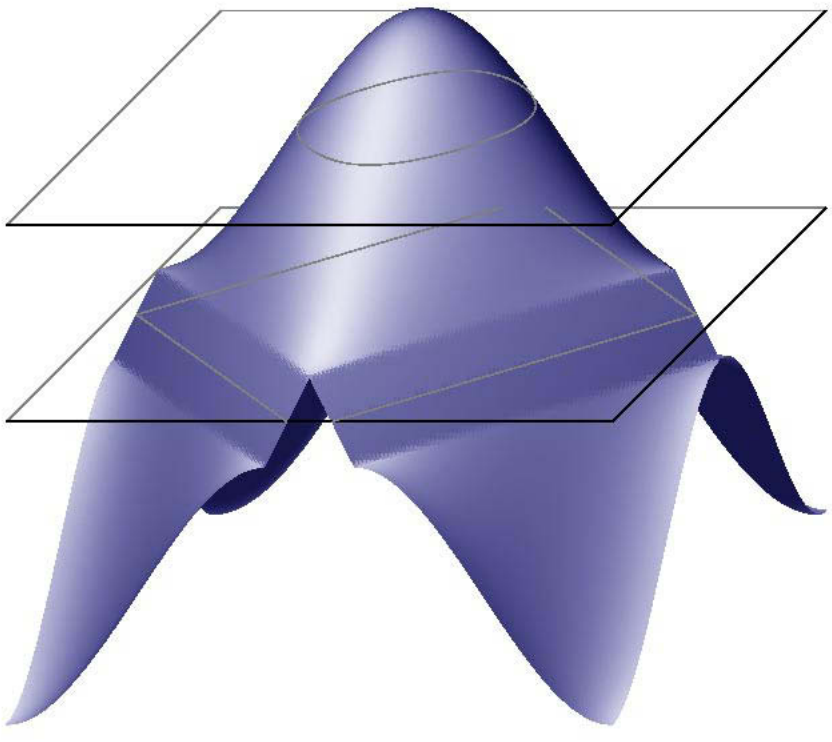
11.2 3D Graphics for Terrains

The next step is the drawing of height contours. The drawing itself is simply done by line segments.
A surface quadrilateral has maximal two edge intersections with a horizontal plane at the height z (*plz* in the program). Maximal two intersections P_1,P_2 of P_a,P_b,P_c and P_d are calculated. A control logic assigns P_1 and P_2 which are connected then by a line.
The problem of arranging all line segments to closed paths is not yet solved. This would require a sorting, by finding connected end points and additionally the insertion of boundary segments for closing open paths.

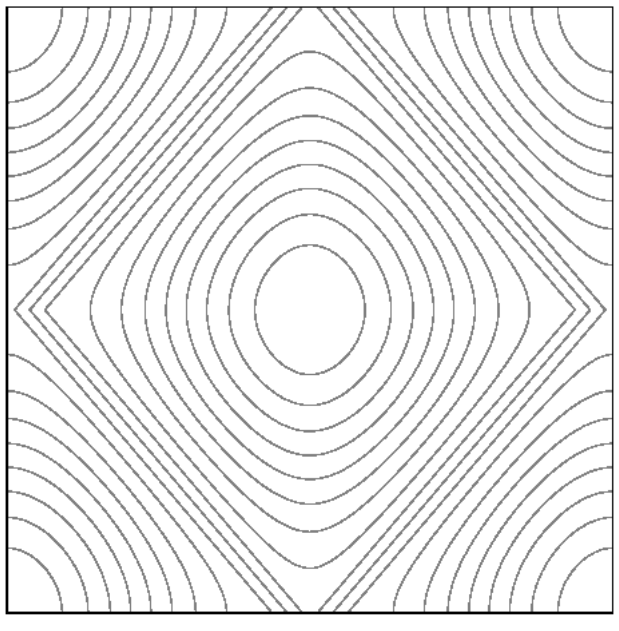


Best view for TIFFs : zoom 100%

Elliptic spots



Elliptic spots



11.3 3D Graphics for Terrains

```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox:  0 0 808 808
%%Creator:      Gernot Hoffmann
%%Title:        Math3D-Contour
%%CreationDate: Dec.24,2002

% Disable setpagedevice
 /setpagedevice {pop} bind def

% A function z=z(x,y) is defined for x=-1..+1, y=-1..+1
% The value z is expected approximately in z=-1..+1

% The graph z(x,y) is shown by a cabinet projection
% Visibility by drawing from back to the front
% Lambert/Phong light model
% Wireframe or rendering
% Height contours, one or two in Cabinet view
% Height map in top view

%-Variables
/mm {2.834646 mul} def

/sx  80 mm def                  % Length -sx..+sx
/sy  80 mm def
/bx 280 mm def                  % Bounding box
/by 280 mm def

/x0 140 mm def                  % Center
/y0 140 mm def

/viw  5 def                     % 2: 2D height map contours at pl2..-pl2
                                % 3: 3D cabinet view
                                % 4: 3D cabinet view with one height contour
                                % 5: 3D cabinet view with two height contours

/rnd  2 def                     % For viw>2:
                                % 1: Wireframe on black background, np=20
                                % 2: Rendered  on white background, np=50..100

/np  10 def                     % function    2np divisions for x,y=-1..+1
/nz   10 def                    % height map  2nz height contours

/ang  45 def                    % y-axis angle,  default 45¦
/scy 0.5 def                    % y-axis length, default 0.5
/pl1 0.00 def                   % Plane z-value, first  plane
/pl2 0.60 def                   % Plane z-value, second plane, start for viw=2

/phg  1 def                     % 1:   Lambert
                                % 2,3  Phong

/lcx  0 def                     % Light position
/lcy -2 def
/lcz  2 def

/env 0.3 def                    % environmental light

/H 240 360 div def              % HSB color order system
/S 1.0 def
/B 1.0 def

/sel 2 def                      % Function selector

%-Preparations

/nn np neg def
/n2 np 2 mul 1 add def
/dxy 1 np div def
/dz  1 nz div def

/kxy ang cos scy mul def
/kyy ang sin scy mul def

/za1 n2 array def
/za2 n2 array def

/plz 0 def
/xf  0 def /yf  0 def /zf  0 def /xd  0 def /yd  0 def
/z11 0 def /z12 0 def /z21 0 def /z22 0 def /dnx 0 def /dny 0 def
/dlx 0 def /dly 0 def /dlz 0 def /cal 0 def /nrm 0 def
/x1 0 def /y1 0 def /x2 0 def /y2 0 def
/x3 0 def /y3 0 def /x4 0 def /y4 0 def
/ck 0 def /ca 0 def /cb 0 def /cc 0 def /cd 0 def
%-Procedures
```

11.4 3D Graphics for Terrains

```
/BoundBox
{ 0 setgray
 /lw 0.2 mm def lw setlinewidth
  lw lw moveto
  bx lw sub 0 rlineto
  0 by lw sub rlineto
  bx neg lw add 0 rlineto
  closepath
  rnd 1 eq {fill   } if
  rnd 2 eq {stroke } if
} def

/FuncBox
{ 0 setgray
 /lw 0.2 mm sx div def lw setlinewidth
  -1 -1 moveto
   2  0 rlineto
   0  2 rlineto
  -2  0 rlineto
  closepath
  stroke
} def

/Func
{ sel 1 eq % a hill
   {/zf 1 xf dup mul yf dup mul add 20 mul 1 add div 0.1 sub def} if
  sel 2 eq % round spot   {/zf xf 180 mul cos yf 180 mul cos add 0.5 mul def} if
  sel 3 eq % elliptic spot simple
   {xf yf dup 5 mul 8 div mul exch dup mul exch add sqrt 1 exch sub /zf exch def} if
  sel 4 eq % elliptic spot, by Helge Blischke, Adobe Forum PS
   { xf yf exch
     abs exch abs 2 copy 0.85 mul add 0.85 lt
     { exch 0.85 div
       dup dup mul exch 2 mul 3 sub mul exch
       dup dup mul exch 2 mul 3 sub mul add 0.85 mul 1 add }
     { 2 copy 0.85 mul add 1 gt
     { 1 sub exch 1 sub 0.85 div dup dup mul exch 2 mul 3 add mul exch
       dup dup mul exch 2 mul 3 add mul add 0.85 mul 1 sub }
     { 0.85 mul add 2 mul neg 1 add 0.85 add } ifelse
     } ifelse /zf exch def } if
} bind def

/Cabin % cabinet projection
{ /xd xf kxy yf mul add def
  /yd zf kyy yf mul add def
} bind def

/LineFront
{  rnd 1 eq {0.5 setgray} if
   rnd 2 eq {0.0 setgray} if
   0.3 mm sx div setlinewidth
  /zf plz def
  /yf -1 def /xf -1 def Cabin xd yd moveto /xf 1 def Cabin xd yd lineto stroke
  /yf -1 def /xf  1 def Cabin xd yd moveto /yf 1 def Cabin xd yd lineto stroke
} def

/LineBack
{  rnd 1 eq {0.2 setgray} if
   rnd 2 eq {0.5 setgray} if
   0.3 mm sx div setlinewidth
  /zf plz def
  /yf -1 def /xf -1 def Cabin xd yd moveto /yf 1 def Cabin xd yd lineto stroke
  /yf  1 def /xf -1 def Cabin xd yd moveto /xf 1 def Cabin xd yd lineto stroke
} def

/Light
% Lambert/Phong light model
% cal=cosine of angle between normal vector and light direction
{ cal 0 lt {/cal 0 def } if
  phg 1 sub { /cal cal dup mul def } repeat
 /cal cal 1 env sub mul env add 0.95 mul def
  H S 1 cal sub mul cal sethsbcolor
} bind def
/RenderF
```

11.5 3D Graphics for Terrains

```
% Draw 3D Cabinet view with 0,1 or 2 height contours
{ 0.15 mm sx div setlinewidth
 /yf +1 def
 /xf -1 def
 /i 0 def
  nn 1 np
 { pop
   Func za1 i zf put
  /i  i 1 add def
  /xf xf dxy add def
 } for
 /yf +1 dxy sub def
  np -1 sub -1 nn 2 add
  {/xf -1 def
   /i 0 def
    nn 1 np
   { pop
     Func za2 i zf put
    /xf xf dxy add def
    /i i 1 add def
   } for
  /xf -1 def
  /i 0 def
   nn 1 np 1 sub
{ pop
  newpath
 /z11 za1 i get def
 /z12 za1 i 1 add get def
 /z21 za2 i get def
 /z22 za2 i 1 add get def
 /zf z11 def /yf yf dxy add def Cabin xd yd moveto
 /zf z12 def /xf xf dxy add def Cabin xd yd lineto
 /zf z22 def /yf yf dxy sub def Cabin xd yd lineto
 /zf z21 def /xf xf dxy sub def Cabin xd yd lineto
  closepath
 /dnx z12 z11 sub z22 z21 sub add 0.5 mul neg def
 /dny z21 z11 sub z22 z12 sub add 0.5 mul def
 /dlx lcx xf sub def
 /dly lcy yf sub def
 /dlz lcz zf sub def
 % cos(alf)=(dnx.dlx)/Sqrt(norm(dnx)*norm(dlx))
 /cal dnx dlx mul dny dly mul add dxy dlz mul add def
 /nrm dnx dup mul dny dup mul add dxy dup mul add
      dlx dup mul dly dup mul add dlz dup mul add mul sqrt def
 /cal cal nrm div def
 /z11 z11 z12 add z21 add z22 add 0.25 mul def
  z11 plz lt {/cal cal 0.03 sub def} if % make darker if z<plz
  Light
  rnd 1 eq {stroke} if
  rnd 2 eq {fill  } if
 /i i 1 add def
 /xf xf dxy add def
 } for
/yf yf dxy sub def
   za2 za1 copy
  } for
} bind def

/RenderC
```

11.6 3D Graphics for Terrains

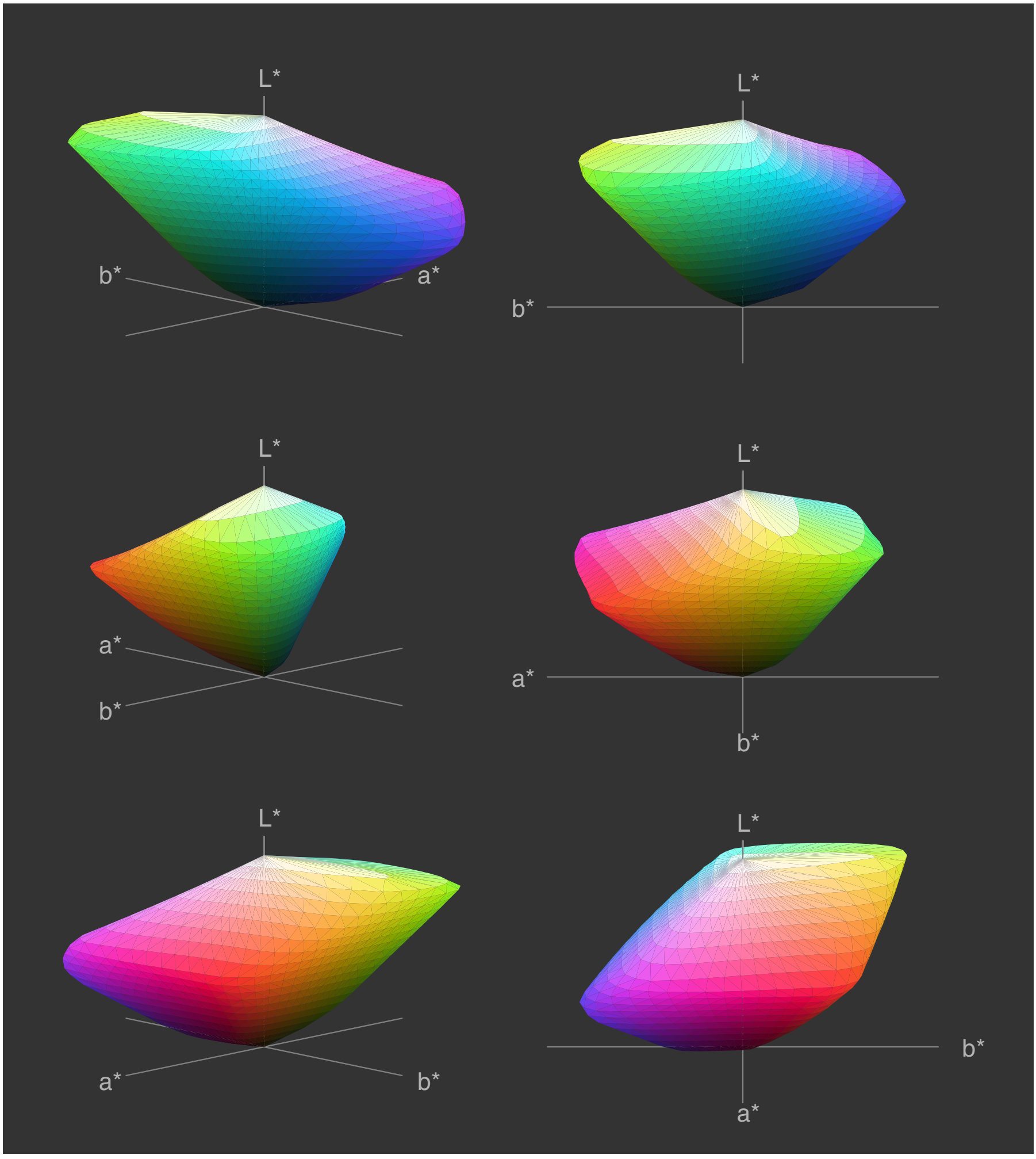
```
% Draw height contour for plane at plz, Cabinet or top view
{ 0.15 mm sx div setlinewidth
  0.5 setgray
 /yf +1 def
 /xf -1 def
 /i 0 def
  nn 1 np
 { pop
   Func za1 i zf put
  /i  i 1 add def
  /xf xf dxy add def
 } for
 /yf +1 dxy sub def
  np -1 sub -1 nn 2 add
  {/xf -1 def
   /i 0 def
    nn 1 np
   { pop
     Func za2 i zf put
    /xf xf dxy add def
    /i i 1 add def
   } for
   /xf -1 def
   /i 0 def
    nn 1 np 1 sub
  { pop
    newpath
   /z11 za1 i get def
   /z12 za1 i 1 add get def
   /z21 za2 i get def
   /z22 za2 i 1 add get def
   /ck 0 def
   /ca false def /cb false def /cc false def /cd false def
    z11 plz le z12 plz ge and z11 plz ge z12 plz le and or
  {/x1 xf dxy plz z11 sub z12 z11 sub div mul add def /y1 yf dxy add def /ck ck 1 add def /ca true def } if
    z21 plz le z22 plz ge and z21 plz ge z22 plz le and or
  {/x2 xf dxy plz z21 sub z22 z21 sub div mul add def /y2 yf def         /ck ck 1 add def /cb true def } if
    ck 2 lt
  { z21 plz le z11 plz ge and z21 plz ge z11 plz le and or
  {/y3 yf dxy plz z21 sub z11 z21 sub div mul add def /x3 xf def         /ck ck 1 add def /cc true def } if
  } if
    ck 2 lt
  { z22 plz le z12 plz ge and z22 plz ge z12 plz le and or
  {/y4 yf dxy plz z22 sub z12 z22 sub div mul add def /x4 xf dxy add def /ck ck 1 add def /cd true def } if
  } if
    ck 2 ge
   { ca cc and { /x2 x3 def /y2 y3 def } if
     ca cd and { /x2 x4 def /y2 y4 def } if
     cb cc and { /x1 x3 def /y1 y3 def } if
     cb cd and { /x1 x4 def /y1 y4 def } if
     cc cd and { /x1 x3 def /y1 y3 def /x2 x4 def /y2 y4 def} if
% Height map / Cabin
    viw 2 eq
    { x1  y1 moveto
      x2  y2 lineto
      stroke }
     { x1  kxy y1 mul add plz kyy y1 mul add moveto
       x2  kxy y2 mul add plz kyy y2 mul add lineto
       stroke } ifelse
    } if
    /i i 1 add def
    /xf xf dxy add def
    } for
   /yf yf dxy sub def
    za2 za1 copy
   } for
} bind def
```

11.7 3D Graphics for Terrains

```
%-Begin

false setstrokeadjust

BoundBox
x0 y0 translate
sx sy scale

viw 2 eq
{
 /plz pl2 dz sub def
{ RenderC
 /plz plz dz sub def
  plz pl2 neg lt {exit} if
} loop
FuncBox
} if

viw 3 eq
{
plz pl1 def
LineBack
RenderF
LineFront
} if

viw 4 eq
{
plz pl1 def
LineBack
RenderF
/plz pl1 def
LineFront
} if

viw 5 eq
{
/plz pl2 def
LineBack
/plz pl1 def
LineBack
RenderF
/plz pl1 def
LineFront
RenderC
/plz pl2 def
LineFront
RenderC
} if

showpage

%%EndDocument
```

12.1 3D Graphics for Convex Objects

Here we draw convex objects using an arbitrary orthogonal perspective projection.
Hidden surface suppression is achieved by backface culling.
Each quadrilateral is represented by two triangles. There are two possibilities how to tesselate a quadrilateral into two triangles. We find the convex alternative (though the object is not strictly convex) by analyzing the topology of the quadriliteral. This is not explained or used in the sample code. Programmed is a crude approximation.
A surface element is shown if its normal points into the direction of the center of projection. The mathematics for the projection are described in [10] .
The example shows some 3D views of the CIELab color space for the Rec.709 RGB reference system.



12.2 3D Graphics for Convex Objects

```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox:  0 0 808 808
%%Creator:      Gernot Hoffmann
%%Title:        Math3D-Perspective
%%CreationDate: Dec.25,2002

% Disable setpagedevice
 /setpagedevice {pop} bind def

% A convex object in parametric representation x(p1,p2),y(p1,p2),z(p1,p2)
% is defined for p1,p2=-1..+1 in the expected volume x,y,z=-1..+1

% The surface is shown by a parallel or central projection
% Visibility by backface culling
% Lambert/Phong light model
% Wireframe or rendering

%-Variables

/mm {2.834646 mul} def

/sx   80 mm def             % Length -sx..+sx
/sy   80 mm def
/bbx 280 mm def             % Bounding box
/bby 280 mm def

/x0  140 mm def             % Center
/y0  140 mm def

/rnd   2 def                % 1 Wireframe on black background, np=20
                            % 2 Rendered  on white background, np=50..100

/np  100 def                % function 2np divisions for p1,p2=-1..+1

/par   2 def                % 1 Parallel projection
                            %           Isometric par=1, cax=+-1, cay=+-1, caz=+-1
                            % 2 Central  projection


/cax  10 def                % Camera position, not allowed in the viewpoint
/cay -20 def
/caz  10 def
/bet   0 def                % Roll angle, default 0
/zum   1 def                % Zoom, default 1

/vix   0 def                % Viewpoint
/viy   0 def
/viz   0 def

/fru  45 def                % Clipping frustum, default 45 degrees

/lix  +5 def                % Light position
/liy  -5 def
/liz   5 def

/phg   1 def                % 1   Lambert
                            % 2,3 Phong

/env 0.3 def                % environmental light

/H 240 360 div def          % HSB color order system
/S 1.0 def
/B 1.0 def

/sel 1 def                  % Function selector
%-Preparations
```

12.3 3D Graphics for Convex Objects

```
% Camera rotation matrix Modified February 07, 2003
/eps 1E-4 def
/dx cax vix sub def
/dy cay viy sub def
/dz caz viz sub def
/dr dx dup mul dy dup mul add sqrt def
  dx abs eps gt dy abs eps gt or
{/gam dx dy neg atan def }
{/gam 0 def              } ifelse
  dz abs eps gt dr abs eps gt or
{/alf dz neg dr atan def }
{/alf 0 def              } ifelse

/ca alf cos def /sa alf sin def
/cb bet cos def /sb bet sin def
/cg gam cos def /sg gam sin def

/c11 cb cg mul sa sb mul sg mul sub def
/c12 cb sg mul sa sb mul cg mul add def
/c13 ca sb mul neg def
/c21 ca sg mul neg def
/c22 ca cg mul def
/c23 sa def
/c31 sb cg mul sa cb mul sg mul add def
/c32 sb sg mul sa cb mul cg mul sub def
/c33 ca cb mul def

/dis cax vix sub dup mul cay viy sub dup mul add caz viz sub dup mul add sqrt def
 par 2 eq {/zum zum dis mul def } if

/fru fru sin fru cos div abs def

/nn np neg def
/n2 np 2 mul 1 add def
/dp 1 np div def

/xa1 n2 array def
/xa2 n2 array def
/ya1 n2 array def
/ya2 n2 array def
/za1 n2 array def
/za2 n2 array def

/lw  0 def
/xf  0 def /yf  0 def /zf  0 def
/xc  0 def /yc  0 def /zc  0 def
/xd  0 def /yd  0 def
/u   0 def /v   0 def /w   0 def
/z11 0 def /z12 0 def /z21 0 def /z22 0 def /dnx 0 def /dny 0 def
/dlx 0 def /dly 0 def /dlz 0 def
/dnx 0 def /dny 0 def /dnz 0 def
/x11 0 def /y11 0 def /z11 0 def /x12 0 def /y12 0 def /z12 0 def
/x21 0 def /y21 0 def /z21 0 def /x22 0 def /y22 0 def /z22 0 def
/ax  0 def /ay  0 def /az  0 def /bx  0 def /by  0 def /bz  0 def
/cp1 0 def /sp1 0 def /cp2 0 def /sp2 0 def
/cal 0 def /cav 0 def /nrm 0 def /vfr 0 def /eps 0 def
/p1  0 def /p2  0 def

%-Procedures

/Func
{ sel 1 eq % Sphere
  {/cp1 p1 180 mul cos def   % Longitude
   /sp1 p1 180 mul sin def
   /cp2 p2 89.99 mul cos def % Latitude
   /sp2 p2 89.99 mul sin def
   /xf cp2 cp1 mul def
   /yf cp2 sp1 mul def
   /zf sp2 def
  } if
  sel 2 eq % Cone
  {/cp1 p1 180  mul cos def  % Longitude
   /sp1 p1 180  mul sin def
   /xf cp1 1.001 p2 abs sub mul 0.7 mul def
   /yf sp1 1.001 p2 abs sub mul 0.7 mul def
   /zf p2 1.5 mul def
  } if
  } bind def
/Transf % Parallel or central projection
```

12.4 3D Graphics for Convex Objects

```
{ % xf yf zf on the stack expected
  /zc exch caz sub def
  /yc exch cay sub def
  /xc exch cax sub def
  /u  c11 xc mul c12 yc mul add c13 zc mul add def
  /w  c31 xc mul c32 yc mul add c33 zc mul add def
   par 1 eq
 {/xd zum u mul def /yd zum w mul def }
 {/v  c21 xc mul c22 yc mul add c23 zc mul add def
   /vfr v fru mul abs def % frustum clipping
    u abs vfr ge w abs vfr ge or
  {/drw false def}{/xd zum u mul v div def /yd zum w mul v div def} ifelse
 } ifelse
} bind def

/Light
% Lambert/Phong light model
% cal=cosine of angle between normal vector and light direction
{ cal 0 lt {/cal 0 def } if
  phg 1 sub { /cal cal dup mul def } repeat
 /cal cal 1 env sub mul env add 0.95 mul def
  H S 1 cal sub mul cal sethsbcolor
} bind def

/RenderF
% Draw 3D view with backface culling
{ 0.15 mm sx div setlinewidth
 /p1 -1 def
 /p2 -1 def
 /i 0 def
  nn 1 np
 { pop
   Func
      xa1 i xf put          ya1 i yf put          za1 i zf put
  /i  i 1 add def
  /p1 p1 dp add def
 } for
 /p2 -1 dp add def
  nn 1 np 1 sub
   {/p1 -1 def
   /i 0 def
    nn 1 np
   { pop
     Func
     xa2 i xf put          ya2 i yf put          za2 i zf put
    /p1 p1 dp add def
    /i i 1 add def
   } for
   /p1 -1 def
   /i 0 def
    nn 1 np 1 sub
  { pop
   /x11 xa1 i get def
   /y11 ya1 i get def
   /z11 za1 i get def
/x12 xa1 i 1 add get def
/y12 ya1 i 1 add get def
/z12 za1 i 1 add get def
/x21 xa2 i get def
/y21 ya2 i get def
/z21 za2 i get def
/x22 xa2 i 1 add get def
/y22 ya2 i 1 add get def
/z22 za2 i 1 add get def

%Normal vector
/ax x12 x11 sub def
/ay y12 y11 sub def
/az z12 z11 sub def
/bx x21 x11 sub def
/by y21 y11 sub def
/bz z21 z11 sub def
/dnx ay bz mul by az mul sub def
/dny az bx mul ax bz mul sub def
/dnz ax by mul ay bx mul sub def
/nrm 1 dnx dup mul dny dup mul add dnz dup mul add sqrt div def
/dnx dnx nrm mul def
/dny dny nrm mul def
/dnz dnz nrm mul def
%View vector
```

12.5 3D Graphics for Convex Objects

```
 /dvx cax x11 sub def
 /dvy cay y11 sub def
 /dvz caz z11 sub def
 /nrm 1 dvx dup mul dvy dup mul add dvz dup mul add sqrt div def
 /dvx dvx nrm mul def
 /dvy dvy nrm mul def
 /dvz dvz nrm mul def

 %Cosine
 /cav dvx dnx mul dvy dny mul add dvz dnz mul add def

  cav 0 ge rnd 1 eq or
{ newpath
 /drw true def
  x11 y11 z11 Transf xd yd moveto
  x12 y12 z12 Transf xd yd lineto
  x22 y22 z22 Transf xd yd lineto
  x21 y21 z21 Transf xd yd lineto
  closepath
  drw % if
 {
   /dlx lix xf sub def
   /dly liy yf sub def
   /dlz liz zf sub def
   /cal dnx dlx mul dny dly mul add dnz dlz mul add def
   /nrm dlx dup mul dly dup mul add dlz dup mul add sqrt def
   /cal cal nrm div def
    Light
    rnd 1 eq {stroke} if
    rnd 2 eq {fill  } if
    } if
  } if

   /i i 1 add def
   /p1 p1 dp add def
   } for
  /p2 p2 dp add def
   xa2 xa1 copy
   ya2 ya1 copy
   za2 za1 copy
  } for
} bind def

/BoundBox % absolute
{ 0 setgray
 /lw 0.2 mm def lw setlinewidth
  newpath
  lw lw moveto
  bbx lw sub 0 rlineto
  0 bby lw sub rlineto
  bbx neg lw add 0 rlineto
  closepath
  rnd 1 eq {fill   } if
  rnd 2 eq {stroke } if
} def

/Axes % relative
{  rnd 1 eq {0.5 setgray} if
   rnd 2 eq {0.0 setgray} if
   0.3 mm sx div setlinewidth
  /zf 0 def
   0 0 0 Transf xd yd moveto 1 0 0 Transf xd yd lineto
   0 0 0 Transf xd yd moveto 0 1 0 Transf xd yd lineto
   0 0 0 Transf xd yd moveto 0 0 1 Transf xd yd lineto stroke
} def

%-Begin

false setstrokeadjust

BoundBox
x0 y0 translate
sx sy scale

RenderF
Axes

showpage

%%EndDocument
```

13.1 IT8 Target

This is an IT8 target, as used for scanner and printer calibration.
The values are not according to the standards - they were taken from an actual target.
More information in the source code.

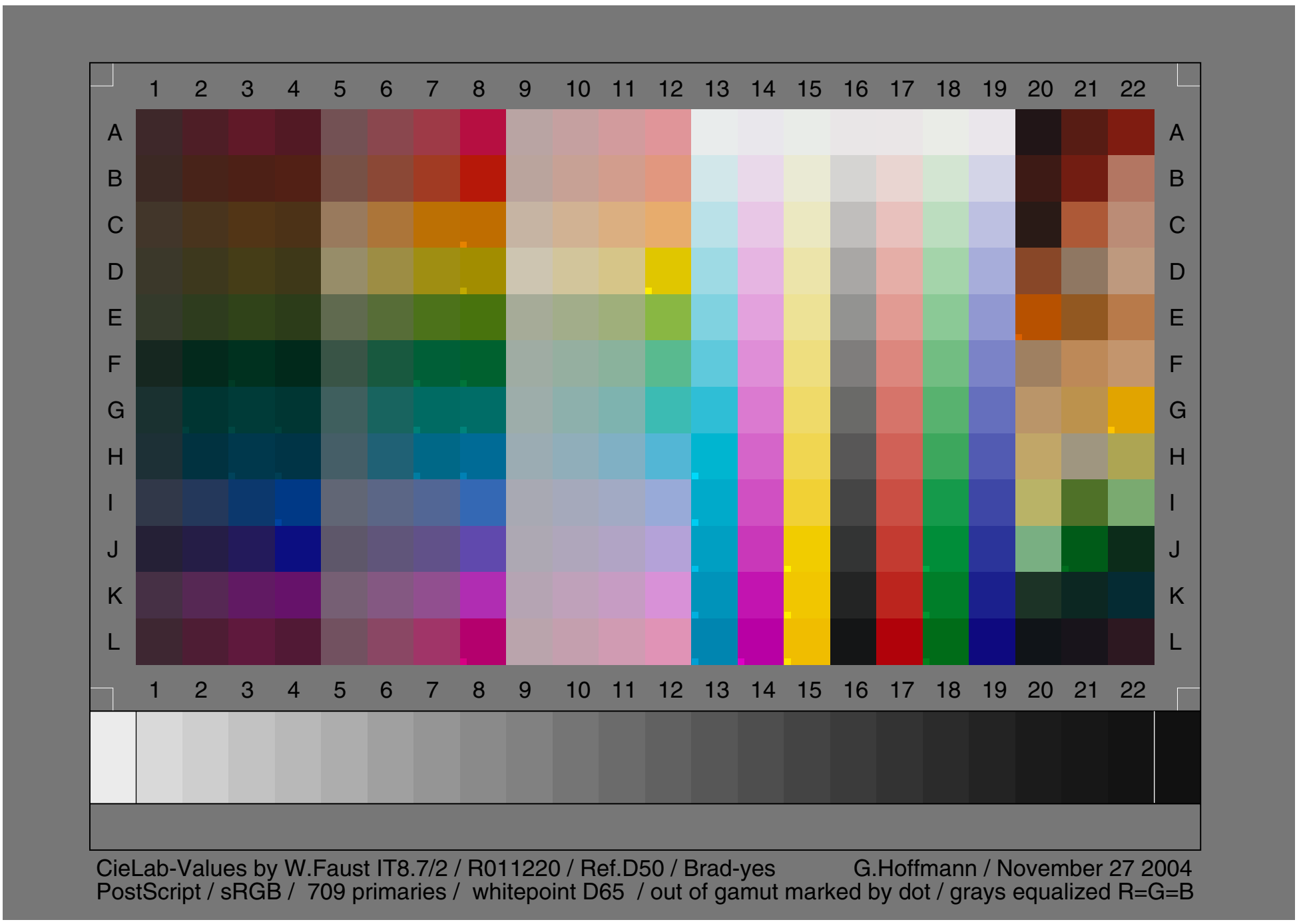
The EPS file for the complete diagram
can be loaded here:

http://www.fho-emden.de/~hoffmann/targetrgb.txt

Please rename then to
targetrgb.eps

The EPS requires the external font
Helvetica

The EPS does not have a TIFF preview



Correct Acrobat view requires Smooth Line Art Off

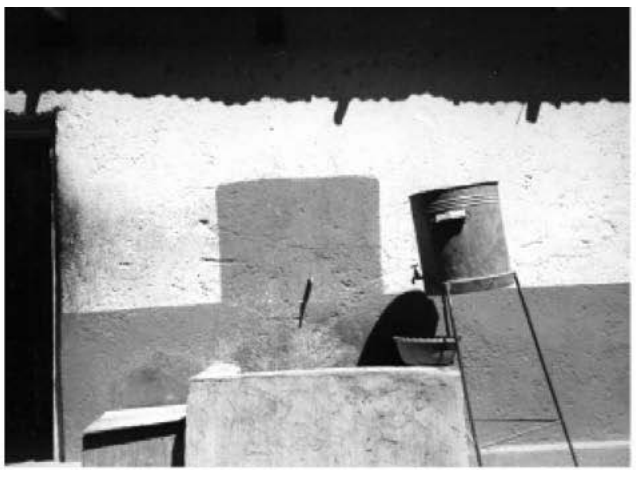
14.1 Image Files

PostScript can hardly be used for image processing, It is far too slow and requires device dependent file buffers. Therefore the example in this chapter cannot be used as an EPS in other programs. Nevertheless - it works with PSAlter [4].

These operations are executed for uncompressed RGB images BMP-24 or BMP-32, named *Src* for source file:

1. Read file header of *Src* and show decoded parameters.
   If OK then continue, otherwise finish.
2. Read file *Src* from disk and save as an intermediate raw RGB file *ImiA* in memory or on disk. Large intermediate files are on the hard disk.
3. Read *ImiA* and show image.
4. Apply grayscale conversion to *ImiA*, save as *ImiB*, rename as *ImiA.*
5. Read *ImiA* and show grayscale image.

BM code	BM
Size/Byte	229446
Image Start	54
Info Size	40
x-Pixels	324
y-Pixels	236
Page	1 0
Bit per Pix	24 0
Compression	0 0



Altogether this example may help to understand the handling of images and files.

14.2 Image Files

```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox:  0 0 1191 842
%%Creator:      Gernot Hoffmann
%%Title:        BmpImage01
%%CreationDate: March 09,2003
%%DocumentNeededResources: font Helvetica
%%EndComments

% Put on A3 landscape

% Show header information
% Read, store and show a named BMP-24  RGB image
% Read, store and show a named BMP-32 ARGB image. Ignore A.
% Manipulate RGB image in buffer and show

/mm {2.834646 mul} def

/buf 0 def /cd1 0 def /cd2 0 def /fsz 0 def /ims 0 def /ifs 0 def
/xpx 0 def /ypx 0 def /pg1 0 def /pg2 0 def /bp1 0 def /bp2 0 def
/cp1 0 def /cp2 0 def /byt 0 def /bmp 0 def
/bh10 0 def /fh 0 def /x1 0 def /x2 0 def /y 0 def /c 0 def
/rowA 0 def /rowB 0 def /bufA 0 def /bufB 0 def /kA 0 def /kB 0 def
/Imod 0 def /gam  0 def /iga  0 def

/ReadHead
{/buf 54 string def
 /bmp true def
% Read 54 header bytes 0..53
  Src buf readstring pop
% Code (BM)
 /cd1      buf 0 1 getinterval def
 /cd2      buf 1 1 getinterval def
  cd1      (B) ne { /bmp false def } if
  cd2      (M) ne { /bmp false def } if
% File Size (Bytes)
 /fsz      buf  2 get
           buf  3 get 256 mul add
           buf  4 get 256 mul 256 mul add
           buf  5 get 256 mul 256 mul 256 mul add def
% Image Start (54)
 /ims      buf 10 get
           buf 11 get 256 mul add
           buf 12 get 256 mul 256 mul add
           buf 13 get 256 mul 256 mul 256 mul add def
% Info Size (40)
 /ifs      buf 14 get
           buf 15 get 256 mul add
           buf 16 get 256 mul 256 mul add
           buf 17 get 256 mul 256 mul 256 mul add def
% x-Pixel
 /xpx      buf 18 get
           buf 19 get 256 mul add
           buf 20 get 256 mul 256 mul add
           buf 21 get 256 mul 256 mul 256 mul add def
% y-Pixel
 /ypx      buf 22 get
           buf 23 get 256 mul add
           buf 24 get 256 mul 256 mul add
           buf 25 get 256 mul 256 mul 256 mul add def
% pages (1,0)
 /pg1      buf 26 get def
 /pg2      buf 27 get def
  pg1      1 ne { /bmp false def } if
  pg2      0 ne { /bmp false def } if
% BitperPixel (24,0 )
 /bp1      buf 28 get def
 /bp2      buf 29 get def
  bp1      24 eq bp1 32 eq or not { /bmp false def } if
% Uncompressed (0,0)
 /cp1      buf 30 get def
 /cp2      buf 31 get def
  cp1      0 ne { /bmp false def } if
  cp2      0 ne { /bmp false def } if
% Bytes per Pixel
 /byt bp1 8 idiv def
} def
/ShowHead
```

14.3 Image Files

```
{/b10 10 string def
 /fh 14 def
 /Helvetica findfont fh scalefont setfont
 /x1 10 mm def
 /x2 50 mm def
 /y  58 mm def
 /dy fh 1.2 mul def
  x1 y moveto                (BM code)      show x2 y moveto cd1 show cd2 show /y y dy sub def
  x1 y moveto (Size/Byte)    show x2 y moveto fsz b10 cvs show  /y y dy sub def
  x1 y moveto (Image Start)  show x2 y moveto ims b10 cvs show  /y y dy sub def
  x1 y moveto (Info Size)    show x2 y moveto ifs b10 cvs show  /y y dy sub def
  x1 y moveto (x-Pixels)     show x2 y moveto xpx b10 cvs show  /y y dy sub def
  x1 y moveto (y-Pixels)     show x2 y moveto ypx b10 cvs show  /y y dy sub def
  x1 y moveto (Page)         show x2 y moveto pg1 b10 cvs show ( ) show pg2 b10 cvs show /y y dy sub
def
  x1 y moveto (Bit per Pix)  show x2 y moveto bp1 b10 cvs show ( ) show bp2 b10 cvs show /y y dy sub
def
  x1 y moveto (Compression)  show x2 y moveto cp1 b10 cvs show ( ) show cp2 b10 cvs show /y y dy sub
def
} def

/ReadImag
{% Read Src,swap RGB->BGR,write Imi
 /rowA xpx byt mul string def
 /rowB xpx 3 mul string def
 /bufA byt string def
 /bufB 3 string def
 byt 3 eq % BMP-24
{1 1 ypx
 { pop
  /kA 0 def
   Src rowA readstring pop
   1 1 xpx
   { pop
    /bufA rowA kA 3 getinterval def
     bufB 0 bufA 2 get put
     bufB 1 bufA 1 get put
     bufB 2 bufA 0 get put
     rowB kA bufB putinterval
    /kA kA 3 add def
   } for
   ImiA rowB writestring
 } for
 } if
 byt 4 eq % BMP-32
{1 1 ypx
 { pop
  /kA 0 def
  /kB 0 def
   Src rowA readstring pop
   1 1 xpx
   { pop
    /bufA rowA kA 4 getinterval def
     bufB 0 bufA 2 get put
     bufB 1 bufA 1 get put
     bufB 2 bufA 0 get put
     rowB kB bufB putinterval
    /kA kA 4 add def
    /kB kB 3 add def
   } for
   ImiA rowB writestring
 } for
 } if
  ImiA flushfile
} bind def

/ImagProc
{% calculate new RGB, range 0..255
  Imod 1 eq % RGB Color identity
  { bufB 0 bufA 0 get put
    bufB 1 bufA 1 get put
    bufB 2 bufA 2 get put } if
  Imod 2 eq % RGB Gray (use evtl. 0.30 0.60 0.10 )
  {/c bufA 0 get 255 div gam exp 0.3333 mul
      bufA 1 get 255 div gam exp 0.3333 mul add
      bufA 2 get 255 div gam exp 0.3333 mul add
      iga exp 255 mul cvi def
    bufB 0 c put
    bufB 1 c put
    bufB 2 c put } if
  Imod 3 eq % RGB Any test
  { bufB 0 255 put
    bufB 1 bufA 0 get put
```

14.4 Image Files

```
/ModiImag
{% Read ImiA,modify, write ImiB, rename ImiA
 /rowA xpx 3 mul string def
 /rowB xpx 3 mul string def
 /bufA 3 string def
 /bufB 3 string def
 /c 1 string def
 /gam 2.2 def
 /iga 1 gam div def
  ImiA 0 setfileposition
 1 1 ypx
 { pop
  /kA 0 def
   ImiA rowA readstring pop
   1 1 xpx
   { pop
    /bufA rowA kA 3 getinterval def
     ImagProc
     rowB kA bufB putinterval
    /kA kA 3 add def
   } for
   ImiB rowB writestring
 } for
  ImiB flushfile
 /ImiA ImiB def
} bind def

/ShowImag
{  /DeviceRGB setcolorspace
<< /ImageType 1
   /Width  xpx
   /Height ypx
   /BitsPerComponent 8
   /ImageMatrix [ xpx 0 0 ypx 0 0 ]
   /Decode      [ 0 1 0 1 0 1 ]
   /DataSource ImiA
>>
image
} def

 /Src (J:\\WEmma\\WEmma295.bmp) (r) file def % small BMP-24
%/Src (J:\\Emma_\\Emma_295.bmp) (r) file def % large BMP-32
%/Src (D:\\WImag\\WImag003.bmp) (r) file def
/ImiA (ImiA) (w+) file def
/ImiB (ImiB) (w+) file def

ReadHead
ShowHead

bmp {
gsave
80 mm 10 mm translate
xpx ypx scale
ReadImag
ShowImag

grestore
220 mm 10 mm translate
xpx ypx scale

% Imod 1 RGB Copy
% Imod 2 RGB Gray
% Imod 3 RGB Any test

/Imod 2 def
ModiImag
ShowImag } if

showpage
```

15.1 Reading ASCII Files

This program shows how to read and to interpret an ASCII file. The header is ignored by dummy reading. Four substrings in each following line are extracted. Three of the four substrings are converted into numbers. The test file is on the next page. The content doesn´t have any meaning. The filter has to be adapted to the expected file structure.

Note: the program can be simplified by using 'token' .
The file targetrgb.txt in chapter 13 shows the application of token.

```
%!PS-Adobe-3.0   EPSF-3.0
%%BoundingBox:   0 0 283 283
%%Creator:       Gernot Hoffmann
%%Title:         Read ASCII File
%%CreationDate: Jan.24, 2003
%%DocumentNeededResources: font Helvetica
%%EndComments

% Read a named ASCII file
% Expected structure
%     String1 String2 String3 String4 (String5 ...)
%     No space before String1
%     Substrings separated by at least one space
%     Spaces after String4 are not necessary but allowed
% Convert and show
%     String1 Number2 Number3 Number4
% Stop conditions
%     End of File or
%     Short string with less than 20 characters, like „End_Data"

/mm {2.834646 mul} def      % points per mm
/spa ( ) def                % ASCII space

/S1 0 def /S2 0 def /S3 0 def /S4 0 def
/N1 0 def /N1 0 def /N3 0 def /N4 0 def
/i  0 def /k  0 def /s  0 def

/FindNums
%  expected buf on stack
{ /dum exch def
  /len dum length def
  /le1 len 1 sub  def
  /i 0 def
  { dum i 1 getinterval spa eq {exit} if
     /i i 1 add def } loop
     /S1 dum 0 i getinterval def
 { dum i 1 getinterval spa ne {exit} if
     /i i 1 add def } loop
     /k i 1 sub def
 { dum i 1 getinterval spa eq {exit} if
     /i i 1 add def } loop
     /s i k sub 1 add def
     /S2 dum k s getinterval def
 { dum i 1 getinterval spa ne {exit} if
     /i i 1 add def } loop
     /k i 1 sub def
 { dum i 1 getinterval spa eq {exit} if
     /i i 1 add def } loop
     /s i k sub 1 add def
     /S3 dum k s getinterval def
 { dum i 1 getinterval spa ne {exit} if
     /i i 1 add def } loop
     /k i 1 sub def
 { dum i 1 getinterval spa eq i le1 ge or {exit} if
     /i i 1 add def } loop
     /s i k sub 1 add def
     /S4 dum k s getinterval def
     /N2 S2 cvr def
     /N3 S3 cvr def
     /N4 S4 cvr def
} bind def
```

15.2 Reading ASCII Files

```
(I:\\PS-Tutor\\testfile.txt) (r) file
/rdf exch def

/buf 100 string def

/fh 10 def
/Helvetica findfont fh scalefont setfont

/x   5 mm def
/y 100 mm def

%—dummy read 14 header lines
mark
1 1 14
{ rdf buf readline
} for
cleartomark

%—read lines, extract and show S1,N2,N3,N4
{ rdf buf readline not {exit} if
     dup /len exch length def
     len 20 le {exit} if
     FindNums
     x y  moveto S1 show
     x 20 mm add y moveto N2 buf cvs show
     x 40 mm add y moveto N3 buf cvs show
     x 60 mm add y moveto N4 buf cvs show
  /y y fh sub def
  } loop

showpage
```

Testfile I:\PS-Tutor\testfile.txt

```
IT8.7/2
ORIGINATOR „XYZ“
DESCRIPTOR „L* a* b* Batch average data (light D65, viewing angle 2)“
MANUFACTURER „XYZ - http://www.xyz.de“
CREATED „January 07, 2002“
PROD_DATE „2001:12“
SERIAL „DIN A4  R011220“
MATERIAL „Kodak Professional Digital 3 - Non-Glossy“
NUMBER_OF_FIELDS 9
BEGIN_DATA_FORMAT
SAMPLE_ID  XYZ_X XYZ_Y XYZ_Z LAB_L LAB_A LAB_B LAB_C LAB_H
END_DATA_FORMAT
NUMBER_OF_SETS 288
BEGIN_DATA
A1        3.07    2.67     2.50    18.64   9.19     -2.55   9.543    44.45
A2        3.75    2.59     2.06    18.32   21.51    0.68    21.52    1.81
A3        5.27    3.20     2.28    20.81   31.07    2.97    31.2     5.46
A4        3.91    2.50     1.90    17.90   25.66    1.56    25.70    3.48
A5        -11.41  10.14    9.66    38.08   12.34    -4.58   13.16    339.63
A6        -13.57  10.23    8.27    38.26   26.16    0.67    26.17    1.46
A7        15.95   -10.36   7.13    38.47   39.70    5.53    40.08    7.93
A8        18.98   -10.02   5.91    37.86   58.65    9.83    59.47    9.51
A9        39.41   39.30    -39.94  68.97   4.84     -10.54  11.60    294.71
A10       41.39   39.30    -38.54
A11       43.43   39.22    36.99
A12       46.14   39.27    35.18
A13       78.55   83.30    90.92
A14       77.08   80.48    90.09
A15       77.89   82.91    87.50
A16       76.15   79.65    86.50
A17       76.21   79.74    86.51
A18       77.94   83.00    87.31
A19       77.02   80.21    89.95
A20       1.03    0.92     0.90
END_DATA
```

16.1 Decoding Character Paths

This program finds the contour paths of a string, here ’G&H’ , saves the parameters as ASCII text file, reads the file and draws the string. The program can be simplified by using ’token’ .
Write contour path to file

```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox:  0 0 595 842
%%Creator:      Gernot Hoffmann
%%Title:        Character Path Write
%%CreationDate: July 07,2003
%%DocumentNeededResources: font Times-Roman
%%+ Helvetica
%%+ AvantGarde-Book
%%+ GillSansMT-Schlbk

% Write file for string path (one or more characters)
% References
% PostScript Language Reference Manual, Addison Wesley
% McGilton+Campione, PostScript by Example, Addison-Wesley
% PSAlter

% Format, ASCII text file, one space following code or number, newline:

% M x0 y0             % moveto
% L x0 y0             % lineto
% C x1 y1 x2 y2 x3 y3 % curveto
% Z                   % closepath

/mm {2.834646 mul} def

/buf 100 string def

/movePF
{ wrf (M) writestring          wrf ( ) writestring
  wrf exch buf cvs writestring wrf ( ) writestring
  wrf exch buf cvs writestring wrf ( \n) writestring } def

/linePF
{ wrf (L) writestring          wrf ( ) writestring
  wrf exch buf cvs writestring wrf ( ) writestring
  wrf exch buf cvs writestring wrf ( \n) writestring } def

/curvPF
{ wrf (C) writestring          wrf ( ) writestring
  wrf exch buf cvs writestring wrf ( ) writestring
  wrf exch buf cvs writestring wrf ( ) writestring
  wrf exch buf cvs writestring wrf ( ) writestring
  wrf exch buf cvs writestring wrf ( ) writestring
  wrf exch buf cvs writestring wrf ( ) writestring
  wrf exch buf cvs writestring wrf ( \n) writestring } def

/closPF
{ wrf (Z) writestring          wrf ( \n) writestring } def

/fh  50 mm def
/AvantGarde-Book findfont fh scalefont setfont
% one or more characters, string in brackets:
 0 0 moveto (G&H) false charpath

% Write data to file
(I:\\PS-Tutor\\charG&H.txt) (w) file
/wrf exch def

/movePF load
/linePF load
/curvPF load
/closPF load
 pathforall
 wrf closefile

% Draw contour
2 setlinewidth
0 setgray
stroke

showpage
```

16.2 Decoding Character Paths

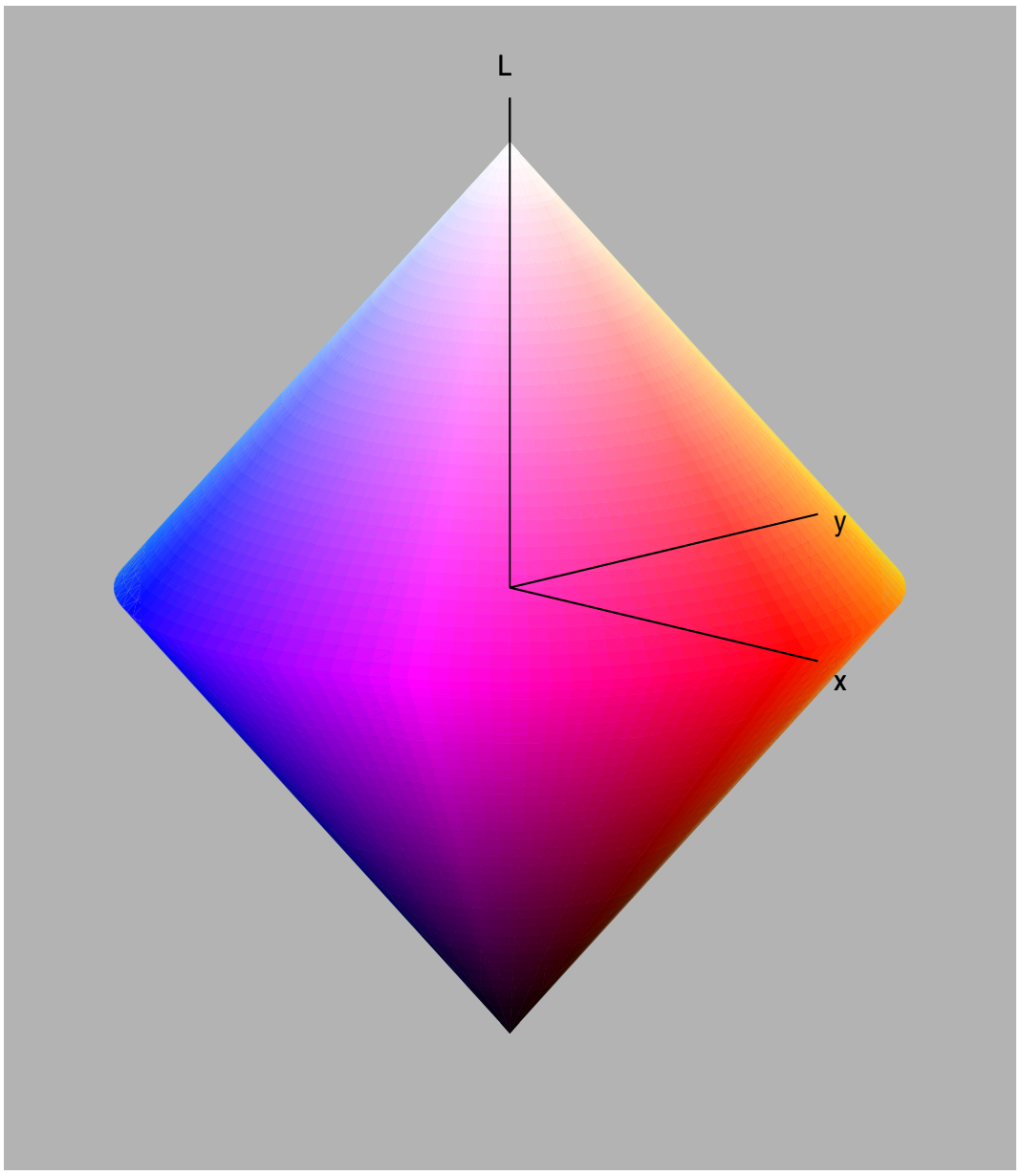
Read contour path from file

```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox:  0 0 595 842
%%Creator:      Gernot Hoffmann
%%Title:        Character Path Read
%%CreationDate: July 07,2003

% Read  file and draw string
% Header as in Write Program

/mm {2.834646 mul} def

/i  0 def /k  0 def
/x0 0 def /y0 0 def /x1 0 def /y1 0 def
/x2 0 def /y2 0 def /x3 0 def /y3 0 def
/spa ( ) def
/buf 100 string def
/cod 1 string def

/Find2Nums
{/i 1 def
 /k i def
 {/i i 1 add def
  buf i 1 getinterval spa eq {exit} if } loop
 /y0 buf k i k sub 1 add getinterval cvr def
 /k i def
 {/i i 1 add def
  buf i 1 getinterval spa eq {exit} if } loop
 /x0 buf k i k sub 1 add getinterval cvr def
} bind def

/Find6Nums
{/i 1 def
 /k i def
 {/i i 1 add def
  buf i 1 getinterval spa eq {exit} if } loop
 /y3 buf k i k sub 1 add getinterval cvr def
 /k i def
 {/i i 1 add def
  buf i 1 getinterval spa eq {exit} if } loop
 /x3 buf k i k sub 1 add getinterval cvr def
 /k i def
 {/i i 1 add def
  buf i 1 getinterval spa eq {exit} if } loop
 /y2 buf k i k sub 1 add getinterval cvr def
 /k i def
 {/i i 1 add def
  buf i 1 getinterval spa eq {exit} if } loop
 /x2 buf k i k sub 1 add getinterval cvr def
 /k i def
 {/i i 1 add def
  buf i 1 getinterval spa eq {exit} if } loop
 /y1 buf k i k sub 1 add getinterval cvr def
 /k i def
 {/i i 1 add def
  buf i 1 getinterval spa eq {exit} if } loop
 /x1 buf k i k sub 1 add getinterval cvr def
} bind def

/Draw
{ newpath
 {rdf buf readline not {exit} if
 /cod buf 0 1 getinterval def
  cod (M) eq { Find2Nums x0 y0 moveto } if
  cod (L) eq { Find2Nums x0 y0 lineto } if
  cod (C) eq { Find6Nums x1 y1 x2 y2 x3 y3 curveto } if
  cod (Z) eq { closepath stroke newpath } if
 } loop
 closepath
} bind def

% Read data from file
(I:\\PS-Tutor\\charG&H.txt) (r) file
/rdf exch def

20 mm 20 mm translate

1 0 0 setrgbcolor
Draw
rdf closefile
showpage
```

17.1 HLS Color Space

PostScript offers the color order system HSV=HSB (hue, saturation brightness or value). HSV is represented by a single cone.
Data are directly converted into RGB.
This color system is highly doubtful, because 'brightness' is interpreted as Max(R,G,B): 200/0/0 is as 'bright' as 200/70/50.

Much better is the color order system HLS (hue, lightness, saturation).
This is represented by a double cone. H=0...360, L=0..1. S=0..1. Everywhere on the surface we have S=1. For L=0.5 and S=1 we will get the primaries R,G,B and the secondaries C,M,Y.




```
/HLStoRGBf
% HLS Foley. Input H L S on stack.
% Output setrgbcolor
{/sat exch def
 /lig exch def
 /hue exch def
  hue 360 gt {/hue 360 sub def} if
  hue   0 lt {/hue 360 add def} if
  lig 1.0 gt {/lig 1.0 def    } if
  lig 0.0 lt {/lig 0.0 def    } if
  sat 1.0 gt {/sat 1.0 def    } if
  sat 0.0 lt {/sat 0.0 def    } if
  lig 0.5 le
  {/max 1.0 sat add lig mul def}
  {/max lig sat add lig sat mul sub def} ifelse
 /min lig 2 mul max sub def
 /mmm max min sub 60 div def
 /hux hue 120 add def
  1
  { hux 360 gt {/hux hux 360 sub def } if
    hux  60 lt {/col min mmm hux mul add         def exit} if
    hux 180 lt {/col max                         def exit} if
    hux 240 lt {/col min mmm 240 hux sub mul add def exit} if /col min def
  } repeat
 /Red col def
 /hux hue def
  1
  { hux  60 lt {/col min mmm hux mul add         def exit} if
    hux 180 lt {/col max                         def exit} if
    hux 240 lt {/col min mmm 240 hux sub mul add def exit} if /col min def
  } repeat
 /Grn col def
 /hux hue 120 sub def
  1
  { hux   0 lt {/hux hux 360 add def} if
    hux  60 lt {/col min mmm hux mul add         def exit} if
    hux 180 lt {/col max                         def exit} if
    hux 240 lt {/col min mmm 240 hux sub mul add def exit} if /col min def
  } repeat
 /Blu col def
% /Red Red iga exp def % gamma encoding by iga=1/2.2 or sRGB
% /Grn Grn iga exp def
% /Blu Blu iga exp def
  Red Grn Blu setrgbcolor
} bind def
```

18.1 Matrix Library

The matrix library [11] contains a couple of procedures for linear algebra and examples how to use them.

Nomenclature

n	Number of rows / n=1 to ...
m	Number of colums / m=1 to ...
Anm	Matrix with n rows and m colums
Xn	Matrix with n rows and 1 column
S1	Scalar / an array with one element
D1	Determinant / an array with one element
xtxt	Coordinate of decimal point
ytxt	Coordinate of decimal point
name	Headline for matrix
fh	Global font height
tms	Global number of digits after decimal point / tms=0 to 8
tmi	Global number of digits before decimal point / space consumer / tmi=1 to ...
num	Arbitrary number

Outputs of procedures have to be arrays (here for D1). The stack contains references for composite objects which reside in the Virtual Memory. Therefore all scalars (S1, D1) are considered as arrays, with the exception of dimensions like n, p,m.
The linear equation solver and the matrix inversion were tested for up to dimension 50.
The determinant D1 can be out of range for formatted printing.
The procedures were not optimized for speed but for readability (named variables).
The typography was tested only for Helvetica. Other fonts may require some corrections.

LF

Prints at least two linefeeds
Requires fh, ytxt
ytxt = ytxt - (count+2)·fh

```
count LF
n LF
0 LF
```

Shownum

Prints one number formatted
Requires font, fh, tms

```
xtxt ytxt num Shownum
```

MatPrn

Prints matrix Anm formatted
Requires font, fh, tms, tmi

```
xtxt ytxt n m (name) Anm MatPrn
xtxt ytxt n 1 (name) Xn  MatPrn % print as column
xtxt ytxt 1 n (name) Xn  MatPrn % print as row
xtxt ytxt 1 1 (Det)  D1  MatPrn
```

18.2 Matrix Library

MatFi0
Fills matrix as zero matrix
```
n m Mnm MatFi1
```

MatFi1
Fills matrix as identity matrix
Fills rectangular matrix by zeros and upper left identity matrix
```
n m Mnm MatFi1
```

Srand
Initializes pseudo random number generator
Uses the integer number seed = 0 to 2^{14}-1
```
seed Srand
```

Xrand
Puts a new random number on the stack
Range -1 to +1
Period 8192
Requires initialization by Srand
```
Xrand
```

MatFiR
Fills matrix by pseudo random numbers
Requires call of Srand
```
n m Mnm MatFiR
```

MatSca
Multiplies matrix by scalar
Bnm = S1·Anm
```
n m S1 Anm Bnm MatSca
```
sequence changed February 18 2007

MatTrn
Calculates transposed matrix
$Bmn = Anm^T$
```
n m Anm Bmn MatTrn
```

MatAdd
Adds matrices
Cnm = Anm + Bnm
```
n m Anm Bnm Cnm MatAdd
```

MatSub
Subtracts matrices
Cnm = Anm - Bnm
```
n m Anm Bnm Cnm MatSub
```

18.3 Matrix Library

MatMul
Multiplies matrices
Cnm = Anp·Bpm

```
n p m Anp Bpm Cnm MatMul
```

MatLin
Solves linear equations
Ann·Xn = Yn
Delivers Xn and the determinant D1 for Ann
Does not check for ill conditioned system
The first n is ignored

```
n n Ann Xn Yn D1 MatLin
```

MatInv
Calculates inverse matrix
$Bnn = Ann^{-1}$
Delivers Bnn and the determinant D1 for Ann
Does not check for ill conditioned system
The first n is ignored

```
n n Ann Bnn D1 MatInv
```

MatNor
Solves overdetermined linear equations by Gauss Normal algorithm
n>m
n=m can be used for tests
$Anm \cdot Xm = Yn$
$Anm^T \cdot Anm \cdot Xm = Anm^T \cdot Yn$
$Cmm \cdot Xm = Zm$
Delivers Xm and the determinant D1 for Cmm
Does not check for ill conditioned system

```
n m Anm Xm Yn D1 MatNor
```

18.4 Matrix Library

MatAxB3
Cross product in 3D
C3 = A3 x B3
The parameter 3 is ignored
```
3 A3 B3 C3 MatAxB3
```

MatAtB
Dot product
C1 = $An^T \cdot Bn$
```
n An Bn C1 MatAtB
```

MatEuc
Euclidian norm for An
C1 = |An|
```
n An C1 MatEuc
```

MatUni
Normalize An
An = An / |An|
```
n An MatUni
```

MatAng3
Calculates in 3D the angle C1 from A3 to B3 about the axis N3 = A3 x B3
Clockwise is not defined in 3D
Range 0° to 180°
If at least one vector has zero or almost zero length: C1=0 and N3=0
tan(C1) = |A3xB3| / $A3^T \cdot B3$
The parameter 3 is ignored
```
3 A3 B3 C1 N3 MatAng3
```

MatAng2
Calculates in 2D the angle C1 in the xy-plane from A2 to B2 about the z-axis
Top view *counter-clockwise*
Range 0° to 360°
If at least one vector has zero or almost zero length: C1=0
tan(C1) = $(A2 \times B2)_z$ / $A2^T \cdot B2$
The parameter 2 is ignored
```
2 A2 B2 C1 MatAng2
```

18.5 Matrix Library

PolyRoot2
Finds all roots of a quadratic polynomial (quadratic equation)
$f(x) = a_0 + a_1 x + a_2 x^2 = 0$
Two, one or no solution [12]
The case is more difficult than expected
RI contains the real and the imaginary parts of roots
Fg indicates the type of the roots: 0, 1, 2 (real) or -2 (conjugate complex)
The parameter 2 is ignored
```
2 A3 RI Fg PolyRoot2
```

PolyRootN
Finds all roots of a degree n polynomial
$f(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n = 0$
An is the coefficient array (n+1 real numbers, a_n can be zero)
RI contains the real and the imaginary parts of roots
Fg indicates the type of the roots: 0, 1, 2 (real) or -2 (conjugate complex)
Solution by *Newton-Bairstow* [13]
```
n An RI Fg PolyRootN
```

MatEig
Finds all eigenvalues of the matrix Mnn. See Remarks and [15].)
Uses the characteristic polynomial by the *Faddejev* method [14]
The first parameter n is ignored
Fg is used as above
Eva contains the eigenvalues by real and imaginary parts (structure in the source code)
Finds all eigenvectors of the matrix Mnn (see Remarks)
Eve contains the eigenvectors by real and imaginary parts (structure in the source code)
```
n n Mnn Eva Fg Eve MatEig
```

MatRes
Calculates the complex residuals for all eigenpairs (λ_k, n_k) of the Matrix Mnn.
$R_{kr}+i \cdot R_{ki} = [M - (\lambda_{kr}+i \cdot \lambda_{ki}) \cdot I] n_k$
Used for tests
```
n n Mnn Eva Eve Res MatRes
```
MatEulA
Calculates the rotation matrix for three Euler angles (aircraft sequence).
The source code contains some advanced examples. See [15] .
```
psi the phi R41 MatEulA
```

Remarks
The rootfinder is not bad, but PostScript uses normally single precision for real numbers. Roots with higher multiplicity cannot be found accurately. Eigenvectors for the case of multiple roots cannot be found at all (zero vector outputs). For details see [15].

18.6 Matrix Library

Example / Matrix Inversion

```
/TestMatInv
% Inverse
{/n 6 def
/n2 n dup mul def
/Ann n2 array def
/Bnn n2 array def
/Cnn n2 array def
/Xn  n array def
/Yn  n array def
/d1  1 array def
% Fill matrix by random values
 0 Srand
 n n Ann MatFiR
xtxt ytxt n n (Ann) Ann MatPrn
n 2 add LF
n n Ann Bnn d1 MatInv
xtxt ytxt n n (Bnn Inverse-Gauss) Bnn MatPrn
n 2 add LF
xtxt ytxt 1 1 (Determinant) d1 MatPrn
3 LF
n n n Ann Bnn Cnn MatMul
xtxt ytxt n n (Ann*Bnn) Cnn MatPrn
} def
```

```
Ann
   -0.988953     0.999573     0.922668    -0.997009    -0.458679     0.979065
   -0.789246    -0.853455    -0.475281    -0.025818    -0.673035     0.180725
    0.711243     0.734924    -0.978699     0.855530     0.850891     0.011292
    0.043762    -0.079041    -0.306335     0.553284     0.144348     0.127014
    0.989563    -0.889099    -0.926941     0.223694     0.488586     0.434143
    0.579895     0.960998    -0.059265    -0.726990     0.414856    -0.911072

Bnn Inverse-Gauss
   -4.950001     0.500001    15.100004   -48.700012    -4.150002   -13.800003
   -2.033334     0.500000     7.600001   -22.700004    -2.400001    -6.300001
    0.400000    -1.000000    -2.200001     5.400002     0.300000     1.100001
   -1.733334     0.000000     4.700000   -13.900004    -1.800000    -4.600000
    8.100003    -2.000001   -25.300004    80.600021     7.700003    22.900005
   -0.250000     0.000000     2.500000    -7.500000    -0.250000    -2.500000

Determinant
    0.058594

Ann*Bnn
    1.000000     0.000000    -0.000002     0.000004     0.000001     0.000001
   -0.000000     1.000000     0.000000    -0.000004    -0.000000    -0.000001
    0.000000     0.000000     1.000000     0.000000    -0.000000     0.000000
    0.000000    -0.000000    -0.000000     1.000000     0.000000     0.000000
    0.000001     0.000000     0.000000     0.000000     0.999999     0.000000
    0.000000     0.000000     0.000000     0.000004    -0.000000     1.000001
```

18.7 Matrix Library

Example / Eigenvalues and Eigenvectors

```
/TestMatEig
{/n 4 def
 /nq n dup mul def
 /n2 n 2 mul def
 /Ann [ 1 -1  0  0
        1  1  0  0
        0  0  1  0
        0  0  0  2 ] def
 /d1 1 array def
 /Flg  n array def
 /Eva n2 array def
 /Eve n2 n mul array def
 /Cnn nq array def
 /Res nq 2 mul array def
 Ann Cnn copy
 n n Ann Eva Flg Eve MatEig % overwrites Ann ???
 Cnn Ann copy
 xtxt ytxt n  n (Ann) Ann MatPrn
 n LF
 xtxt ytxt n  2 (Eigenvalues Re-Im) Eva MatPrn
 n LF
 xtxt ytxt n n2 (Eigenvectors Re1-Im1 Re2-Im2 ...) Eve MatPrn
 n LF
% Residuals
 n n Ann Eva Eve Res MatRes
 xtxt ytxt n n2 (Residuals) Res MatPrn
} def % TestMatEig
```

```
Ann
 1.000000  -1.000000   0.000000   0.000000
 1.000000   1.000000   0.000000   0.000000
 0.000000   0.000000   1.000000   0.000000
 0.000000   0.000000   0.000000   2.000000

Eigenvalues Re-Im
 2.000000   0.000000
 1.000000   0.000000
 1.000000   1.000000
 1.000000  -1.000000

Eigenvectors Re1-Im1 Re2-Im2 ...
 0.000000   0.000000   0.000000   0.000000   1.000000   0.000000   1.000000   0.000000
 0.000000   0.000000   0.000000   0.000000   0.000000  -1.000000   0.000000   1.000000
 0.000000   0.000000   1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
 1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

Residuals
 0.000000   0.000000   0.000000   0.000000  -0.000000   0.000000  -0.000000  -0.000000
 0.000000   0.000000   0.000000   0.000000   0.000000  -0.000000   0.000000   0.000000
 0.000000   0.000000  -0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
 0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
```

19.1 References

[1] PostScript Language Reference / third edition
Addison-Wesley
Boston, San Francisco ... / 2002

[2] PostScript Language Tutorial and Cookbook
Addison-Wesley
Reading, Massachusetts ... / 1999

[3] Henry McGilton + Mary Campione
PostScript by Example
Addison-Wesley
Reading, Massachusetts ... / 1998

[4] Quite Software Ltd
sales@quite com
http://www.quite.com

[5] References for Color Science
http://www.fho-emden.de/~hoffmann/colcie290800.pdf

[6] Maxim Shemanarev´s Interpolation
http://www.antigrain.com/agg_research/bezier_interpolation.html

[7] Spline Pascal Source Code
http://docs-hoffmann.de/spline04112001.pdf

[8] Spline Theory
http://docs-hoffmann.de/masspoint09092002.pdf

[9] Bézier Curves
http://docs-hoffmann.de/bezier18122002.pdf

[10] Euler Angles and Projections
http://docs-hoffmann.de/euler26112001.pdf

[11] Matrix Library
http://docs-hoffmann.de/matrixlib.txt
Rename as *.eps

[12] G.Hoffmann
Solutions for a Quadratic Equation
http://docs-hoffmann.de/quadequ04062002.pdf

[13] H.R.Schwarz
Numerische Mathematik
B.G.Teubner
Stuttgart / 1993

19.2 References

[14] D.K.Faddejev und W.N.Faddejewa
Numerische Methoden der linearen Algebra
R.Oldenbourg Verlag
München, Wien / 1979

[15] G.Hoffmann
Faddejev Algorithm for Eigenvalues and Eigenvectors
http://docs-hoffmann.de/faddejev22022007.pdf

This doc
http://docs-hoffmann.de/pstutor22112002.pdf

February 01 / 2013:
Converted from PageMaker to InDesign
May have caused minor layout bugs

Gernot Hoffmann
November 22 / 2002 + February 01 / 2013
Website
Load Browser / Click here