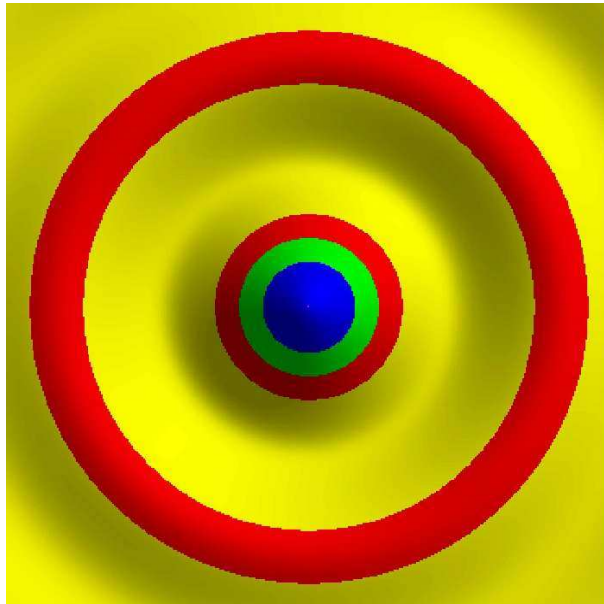


Quick Reference of eps-Toolkit 2.1 for Octave and MATLAB(R)



www.epstk.de

Stefan Müller
stefan.mueller@fgan.de

FGAN, Wachtberg Werthhoven, April 8, 2005

Contents

1	Introduction	7
2	Global Parameters	9
2.1	The toolkit and the parameter	9
2.2	Handling of parameter	9
2.3	Default Values	9
3	User Functions	17
3.1	Basic Functions	17
3.1.1	ebbox.m	17
3.1.2	ebitmap.m	17
3.1.3	eclose.m	18
3.1.4	eopen.m	18
3.1.5	eview.m	19
3.2	Axes Functions	20
3.2.1	eaxes.m	20
3.2.2	eaxespol.m	21
3.2.3	eaxis.m	21
3.3	Grid Functions	22
3.3.1	egrid.m	22
3.3.2	egridpol.m	23
3.4	Plot Functions	24
3.4.1	ebar.m	24
3.4.2	econtour.m	24
3.4.3	eerrbar.m	25
3.4.4	epie.m	25
3.4.5	eplot.m	26
3.4.6	epolar.m	27
3.5	Images Functions	28
3.5.1	ebright.m	28
3.5.2	econtra.m	29
3.5.3	eidx2rgb.m	29
3.5.4	eimage.m	29
3.5.5	eimagesc.m	30
3.5.6	eimg2txt.m	31
3.5.7	eimgmask.m	31
3.5.8	eimgmix.m	32
3.5.9	eimgread.m	32

3.5.10	eimgrot.m	32
3.5.11	eimgview.m	33
3.5.12	eimgwrit.m	33
3.5.13	eimgzoom.m	34
3.5.14	ejpgread.m	34
3.5.15	epolari.m	34
3.5.16	epolaris.m	35
3.5.17	ergb2idx.m	35
3.5.18	ergbsplitt.m	36
3.5.19	eshadoi.m	36
3.5.20	eshadois.m	36
3.5.21	eshadoix.m	37
3.6	Line Functions	38
3.6.1	eellipse.m	38
3.6.2	eframe.m	38
3.6.3	elines.m	39
3.6.4	epline.m	40
3.7	Text Functions	41
3.7.1	eparam.m	41
3.7.2	etext.m	41
3.7.3	etxtbox.m	42
3.7.4	etxtpos.m	42
3.7.5	etxtread.m	43
3.7.6	etxtwrit.m	43
3.8	Symbol Functions	43
3.8.1	edsymbol.m	43
3.8.2	equiver.m	44
3.8.3	esymbol.m	44
3.9	Table Functions	45
3.9.1	etabdef.m	45
3.9.2	etabgrid.m	45
3.9.3	etabtext.m	46
3.10	Special Functions	47
3.10.1	ecdcover.m	47
3.10.2	ecolors.m	47
3.10.3	efillmat.m	47
3.10.4	egradient.m	48
3.10.5	einflate.m	48
3.10.6	einseps.m	48
3.10.7	eisoline.m	49
3.10.8	ejpglist.m	49
3.10.9	elineip.m	49
3.10.10	emerge.m	50
3.10.11	eplo2win.m	50
3.10.12	eshadow.m	50
3.10.13	esubeps.m	51
3.10.14	ewinsize.m	51

A	53
A.1 Examples	53
A.1.1 edemo1.m	53
A.1.2 edemo2.m	54
A.1.3 edemo3.m	57
A.1.4 edemo4.m	61
A.1.5 edemo5.m	63
A.1.6 edemo6.m	66
A.1.7 edemo7.m	68
A.1.8 edemo8.m	72
A.1.9 edemo9.m	74
A.1.10 edemo10.m	76
A.1.11 edemo11.m	79
A.1.12 edemo12.m	83
A.1.13 edemo13.m	84
A.1.14 edemo14.m	86
A.1.15 edemo15.m	91
A.1.16 edemo16.m	93
A.2 Character Code	95

Chapter 1

Introduction

This EPS-Toolkit is a solution of my big problem that I had 1997. I needed graphical output functions for programs which ran with Octave (a freeware Matlab-clone on Unix(Linux)-systems) and Matlab. But I could not find any tool in the internet. So, I wrote this tool myself.

The toolkit consists of some matlab functions, which generate graphical outputs with postscript commands. To view and print the generated postscript files you can use Ghostscript or Ghostview (freeware for all systems).

Features

- MATLAB(R)-Code(4.3) with graphical output functions runs in Matlab and Octave
- All functions are pure m-files
- Most 2D scientific graphics functions are written
- Modifications and extensions are no problem
- Generated EPS-files are very small and importable with no loss of quality
- WYSIWYG (with Ghostview)
- Freeware (GPL)

Difference to version 2.0

- auto bounding box with ebbox
- new image functions e.g. eimgmix, eimgzoom, ...
- new special functions e.g. emerge, einflate, ...
- more fonts
- many bugs removed in ebitmap, edsymbol, eisoline ...

Requirements

- Matlab $\geq 3.x$ or Octave $\geq 2.x$
- Ghostscript
- Ghostview or other viewer for EPS-Files

Installation

1. Include the epstk-directory in the octave-path or matlab-path
2. Edit file "einit.m" in epstk-directory to set the Parameter "ePath", "eGhostview" and "eGhostscript".

Testing

1. Start matlab or octave
2. Start edemos or the single the samples:demo1, demo2, ... demo16

If all demos works, epstk should be ok.

Thanks

Thanks to Josef Worms, Jörg Heckenbach, Coletta Schumacher and Gerd Krämer.

Chapter 2

Global Parameters

2.1 The toolkit and the parameter

This toolkit needs global variables of MATLAB(R) or octave to get default values for the functions. All global parameter are defined in the file 'einit.m'. Which global parameter are used by a function you can see in the description of the functions after this chapter.

2.2 Handling of parameter

If you want to change any default values, edit the file 'einit.m'. This file is called by function 'eopen' everytime.

You can change a global parameter in a program temporarily, if you have called 'eglobpar' before. If you need the default values again after modifications of the global parameter, you should use the commands 'esavpar' and 'erespar'. 'esavpar' saves all global parameters and 'erespar' restores the last backup.

Please note: There are some parameter you can not change temporarily! They are parameter like 'epsFileName' or 'pageOrientation', which initialize the plot-file, the page and the window with the function 'eopen'. In this case use the parameter of 'eopen'.

2.3 Default Values

Here are the default values of the toolkit (listing of 'einit.m'):

```
%%NAME
%% einit - initialize the global parameters of epstk
%%
%%SYNOPSIS
%% einit
%%
% Path of epstk-mfiles
```

```

ePath='./'; % local directory
%ePath='/usr/share/octave/site/m/octave-epstk/';%or e.g. on a linux system

% Read user-parameter-file (octave on linux system)
%eInitUserFile=[system('echo -n $HOME') './epstkInit.m'];
%if exist(eInitUserFile)~=2
% [userParaText n]=etxtrread([ePath 'einit.m']);
% userParaStartPos=findstr(userParaText(1:1000),'% Parameter for epstk');
% etxtwrit(userParaText(userParaStartPos(2):n),eInitUserFile);
% clear userParaText
%end
%source(eInitUserFile);
%return

% Parameter for epstk
% written by stefan.mueller@fgan.de (C) 2005

% Program filename of Ghostscript
eGhostscript='gs'; %ghostscript for linux
%eGhostscript='"c:/gs/gs7.04/bin/gswin32.exe"'; % e.g. on windows system
%eGhostscript=''; %no ghostscript

% Program filename of postscript-viewer
eGhostview='gv'; %gv for linux
%eGhostview='gv --scale=-2 --media=BBox'; %with scale option of gv-version >3.6 for linux
%eGhostview='"c:/gs/gsview/gsview/gsview32.exe"'; %ghostview for windows
%eGhostview=''; %no ghostview

% common parameters
eFileName='epstkout.eps'; % default eps-outputfile
%eFileName=tmpnam; % default eps-outputfile for a octave-systems

% User unit
eUserUnit='mm'; % or 'cm' or 'inch' or 'inch/72'
if strcmp(eUserUnit,'mm'),eFac=2.834646;
elseif strcmp(eUserUnit,'cm'),eFac=28.34646;
elseif strcmp(eUserUnit,'inch'),eFac=72;
else eFac=1;
end

eFile=0; %fileId of eFileName

% fonts (standard fonts of postscript)
eFonts=[
'Times-Roman % font number 1
'Times-Italic % font number 2
'Times-Bold % font number 3
'Times-BoldItalic % font number 4
'Helvetica % font number 5
'Helvetica-Oblique % font number 6

```

```

'Helvetica-Bold           ';           % font number 7
'Helvetica-BoldOblique    ';           % font number 8
'Courier                  ';           % font number 9
'Courier-Oblique          ';           % font number 10
'Courier-Bold             ';           % font number 11
'Courier-BoldOblique      ';           % font number 12
'Symbol                   ';           % font number 13
'Bookman-Demi             ';           % font number 14
'Bookman-DemiItalic       ';           % font number 15
'AvantGarde-Book          ';           % font number 16
'AvantGarde-BookOblique   ';           % font number 17
'AvantGarde-Demi          ';           % font number 18
'AvantGarde-DemiOblique   ';           % font number 19
'Helvetica-Narrow         ';           % font number 20
'Helvetica-Narrow-Oblique ';           % font number 21
'Helvetica-Narrow-Bold    ';           % font number 22
'Helvetica-Narrow-BoldOblique';       % font number 23
'Palatino-Roman           ';           % font number 24
'Palatino-Italic          ';           % font number 25
'Palatino-Bold            ';           % font number 26
'Palatino-BoldItalic      ';           % font number 27
'NewCenturySchlbk-Roman   ';           % font number 28
'NewCenturySchlbk-Italic  ';           % font number 29
'NewCenturySchlbk-Bold    ';           % font number 30
'NewCenturySchlbk-BoldItalic';        % font number 31
'ZapfChancery-MediumItalic';          % font number 32

% colormaps
eColorMaps=[...
    %0 black->white                get it with ecolords(0)
    0 0.0 0.0 0.0;0 1.0 1.0 1.0;

    %1 red->yellow                  get it with ecolords(1)
    1 0.4 0.0 0.0;1 1.0 0.0 0.0;1 1.0 1.0 0.0;

    %2 violet->blue->yellow->red    get it with ecolords(2)
    2 0.4 0.0 0.4;2 0.0 0.0 1.0;2 0.0 1.0 1.0;2 1.0 1.0 0.0;2 1.0 0.0 0.0;

    %3 blue->green->yellow->red     get it with ecolords(3)
    3 0.0 0.0 0.4;3 0.0 0.0 1.0;3 0.0 1.0 1.0;3 0.0 1.0 0.0;3 1.0 1.0 0.0;
    3 1.0 0.0 0.0;

    %4 black->violet->blue->green->yellow->red get it with ecolords(4)
    4 0.1 0.0 0.1;4 0.4 0.0 0.4;4 0.0 0.0 1.0;4 0.0 1.0 0.0;4 0.0 1.0 1.0;
    4 1.0 1.0 0.0;4 1.0 0.0 0.0;

    %5 green->yellow->red->violet   get it with ecolords(5)
    5 0.0 0.4 0.0;5 0.0 1.0 0.0;5 1.0 1.0 0.0;5 1.0 1.0 0.0;
    5 1.0 0.0 0.0;5 0.5 0.0 0.2;

```

```

%6 white->black->violet->blue->green->yellow->red      get it with ecolords(6)
6 1.0 1.0 1.0;6 0.0 0.0 0.0;6 0.4 0.0 0.4;6 0.0 0.0 1.0;6 0.0 1.0 0.0;
6 0.0 1.0 1.0;6 0.0 1.0 0.0;6 1.0 1.0 0.0;6 1.0 0.0 0.0;

%7 grey->yellow->red                                     get it with ecolords(7)
7 1.0 1.0 0.9;7 1.0 1.0 0.0;7 1.0 0.0 0.0;

%8 white->blue->grey->red->white                         get it with ecolords(8)
8 1.0 1.0 1.0;8 0.2 0.2 1.0;8 0.5 0.5 0.5;8 1.0 0.2 0.2;
8 1.0 1.0 1.0;
];

% page
ePageWidth=210; % mm A3=297 A4=210 A5=148
ePageHeight=297;% mm A3=420 A4=297 A5=210
ePageOrientation=0; % 0=Portrait 1=Landscape 2=Upside-down 3=Seaside
ePageReflection=0; % 1=on 0=off reflect page
eXScaleFac=1; % 1=no resize 0.5=50% reduce 2=200% enlarge
eYScaleFac=1; % 1=no resize 0.5=50% reduce 2=200% enlarge

% window
eWinWidth=180; % mm
eWinHeight=250; % mm
eWinFrameVisible=0; % 1=on 0=off draw frame around window
eWinFrameLineWidth=0.3; % mm
eWinGridVisible=0; % 1=on 0=off draw grid of window
eWinTimeStampVisible=0; % 1=on 0=off print time stamp outside of frame
eWinTimeStampFont=1; % font number 1=TimesRoman select font of time stamp
eWinTimeStampFontSize=1.5; % mm

% plot area
ePlotAreaPos=[40 100]; % x y position of left bottom corner of plot area
ePlotAreaWidth=100; % mm
ePlotAreaHeight=100; % mm
ePlotAreaXValueStart=0; % value range of x-axis
ePlotAreaXValueEnd=100;
ePlotAreaYValueStart=0; % value range of y-axis
ePlotAreaYValueEnd= 100;
ePlotLineNo=0;

% polar plot area
ePolarPlotAreaCenterPos=[90 160]; % x y position of Center of polar plot area
ePolarPlotAreaRadMin=10; % mm
ePolarPlotAreaRadMax=50; % mm
ePolarPlotAreaAngStart=0; % deg, 0=east 90=north 180=west 270=south
ePolarPlotAreaAngEnd=360; % deg, 0=east 90=north 180=west 270=south
ePolarPlotAreaValStart=0; % value range of radius-axis
ePolarPlotAreaValEnd=100;
ePolarPlotLineNo=0;

```

```

ePieSliceNo=0;

% title above plots
ePlotTitleDistance=20; % mm
ePlotTitleFontSize=6; % mm
ePlotTitleText=''; % text string
ePlotTitleTextFont=1; % font number 1=TimesRoman

% grid
eXGridLineWidth=0.1; % mm
eXGridColor=[0 0 0]; % [r g b] [0 0 0]=black [1 1 1]=white
eXGridDash=0.5; % mm 0=solid line >0=dash length
eXGridVisible=0; %
eYGridLineWidth=0.1; % mm
eYGridColor=[0 0 0]; % [r g b] [0 0 0]=black [1 1 1]=white
eYGridDash=0.5; % mm 0=solid line >0=dash length
eYGridVisible=0; % 0=off 1=on

% polar grid
ePolarRadiusGridLineWidth=0.1; % mm
ePolarRadiusGridColor=[0 0 0]; % [r g b] [0 0 0]=black [1 1 1]=white
ePolarRadiusGridDash=1; % mm 0=solid line >0=dash length
ePolarRadiusGridVisible=1; % 0=off 1=on
ePolarAngleGridLineWidth=0.1; % mm
ePolarAngleGridColor=[0 0 0]; % [r g b] [0 0 0]=black [1 1 1]=white
ePolarAngleGridDash=2; % mm 0=solid line >0=dash length
ePolarAngleGridVisible=1; % 0=off 1=on

% axes
eAxesColor=[0 0 0]; % [r g b] [0 0 0]=black [1 1 1]=white
eAxesLineWidth=0.3; % mm
eAxesTicShortLength=1.5; % mm
eAxesTicLongLength=3; % mm
eAxesTicLongMaxN=9; % max. number of long Tics
eAxesValueSpace=1; % mm
eAxesValueFontSize=4; % mm
eAxesLabelFontSize=4; % mm
eAxesLabelTextFont=5; % font number 5=Helvetica
eAxesCrossOrigin=0; % 0=off 1=on 2=on and with arrows

% scale vectors:if start=0 and end=0 then autorange,if step=0 then autoscale
% south axis
eXAxisSouthScale=[0 0 0]; % [start step end]
eXAxisSouthScaleType=0; % 0=linear 1=classes 2=log10
eXAxisSouthValueFormat=-1; % n digits after decimal point,-1=auto
eXAxisSouthValueVisible=1; % 0=off 1=on
eXAxisSouthValuePos=[0 0]; % value positions after drawing of axis
eXAxisSouthLabelDistance=2; % mm label distance from axis
eXAxisSouthLabelText='';
eXAxisSouthVisible=1; % 0=off 1=on

```

```

% north axis
eXAxisNorthScale=[0 0 0]; % [start step end]
eXAxisNorthScaleType=0; % 0=linear 1=classes 2=log10
eXAxisNorthValueFormat=-1; % n digits after decimal point,-1=auto
eXAxisNorthValueVisible=1; % 0=off 1=on
eXAxisNorthValuePos=[0 0]; % value positions after drawing of axis
eXAxisNorthLabelDistance=2; % mm label distance from axis
eXAxisNorthLabelText='';
eXAxisNorthVisible=1; % 0=off 1=on

% west axis
eYAxisWestScale=[0 0 0]; % [start step end]
eYAxisWestScaleType=0; % 0=linear 1=classes 2=log10
eYAxisWestValueFormat=-1; % n digits after decimal point,-1=auto
eYAxisWestValueVisible=1; % 0=off 1=on
eYAxisWestValuePos=[0 0]; % value positions after drawing of axis
eYAxisWestLabelDistance=6; % mm label distance from axis
eYAxisWestLabelText='';
eYAxisWestVisible=1; % 0=off 1=on

% east axis
eYAxisEastScale=[0 0 0]; % [start step end]
eYAxisEastScaleType=0; % 0=linear 1=classes 2=log10
eYAxisEastValueFormat=-1; % n digits after decimal point,-1=auto
eYAxisEastValueVisible=1; % 0=off 1=on
eYAxisEastValuePos=[0 0]; % value positions after drawing of axis
eYAxisEastLabelDistance=6; % mm label distance from axis
eYAxisEastLabelText='';
eYAxisEastVisible=1; % 0=off 1=on

% polar radius axis
ePolarAxisRadScale=[0 0 0]; % [start step end]
ePolarAxisRadScaleType=0; % 0=linear 1=classes 2=log10
ePolarAxisRadValueFormat=-1; % n digits after decimal point,-1=auto
ePolarAxisRadValueVisible=3; % 0=off,1=RadStart on,2=RadEnd on,3=Start+End on
ePolarAxisRadValuePos=[0 0]; % value positions after drawing of axis
ePolarAxisRadVisible=1; % 0=off,1=RadStart on,2=RadEnd on,3=Start+End on

% polar angle axis
ePolarAxisAngScale=[0 0 0]; % [start step end]
ePolarAxisAngValueFormat=-1; % n digits after decimal point,-1=auto
ePolarAxisAngValueVisible=1; % 0=off 1=on
ePolarAxisAngValueAngle=0; % angle positions of values after drawing of axis
ePolarAxisAngVisible=1; % 0=off 1=on

% plot line
ePlotLineColor=[0 0 0]; % [r g b] [0 0 0]=black [1 1 1]=white
ePlotLineDash=0; % mm 0=solid line,>0=dash length,<0=fill line,'abc'=symbol abc
ePlotLineWidth=0.3; % mm

```

```

ePlotLineInterpolation=0; % 0=off 1=on

% plot legend
ePlotLegendPos=[-15 -20]; % position relativ to left bottom corner of plot area
ePlotLegendFontSize=4; % mm
ePlotLegendDistance=100; % in percent, depend on ePlotLegendFontSize
ePlotLegendTextFont=1; % font number 1=TimesRoman

% image
eImageDefaultColorMap=0; % number of default map of eColorMaps
eImageFrameVisible=0; % 0=off 1=on

%image legend
eImageLegendPos=[0 -25]; % position relativ to left bottom corner of plot area
eImageLegendWidth=0; % mm 0=ePlotAreaWidth
eImageLegendHeight=5; % mm
eImageLegendScale=[0 0 0]; % [start step end]
eImageLegendScale=[0 0 0]; % [start step end]
eImageLegendScaleType=0; % 0=linear 1=classes 2=log10
eImageLegendValueFormat=-1; % n digits after decimal point, -1=auto
eImageLegendValueVisible=1; % 0=off 1=on
eImageLegendValuePos=[0 0]; % value positions after drawing of axis
eImageLegendLabelDistance=2; % mm
eImageLegendLabelText='';
eImageLegendVisible=1; % 0=off 1=on

% parameter
eParamPos=[30 65]; % absolut position of window
eParamFontSize=4; % mm
eParamLineDistance=100; % in percent, depend on eParamFontSize
eParamTextValueDistance=100; % in percent, depend on eParamFontSize
eParamText='';
eParamTextFont=3; % font number 1=TimesRoman
eParamValue='';
eParamValueFont=11; % font number 9=Courier

% line
eLineWidth=0.3; % mm
eLineColor=[0 0 0]; % [r g b] [0 0 0]=black [1 1 1]=white
eLineDash=0; % mm 0=solid line >0=dash length

% text
eTextFont=1; % font number 1=TimesRoman
eTextFontSize=4; % mm
eTextPos=[30 eWinHeight-eTextFontSize]; % initial position is left top of window
eTextColor=[0 0 0]; % [r g b] [0 0 0]=black [1 1 1]=white
eTextAlignment=1; % 1=right 0=center -1=left
eTextRotation=0; % in deg
eTextLimitWord=' '; % character to limit words
eTextLimitPara=setstr(10); % character to limit paragraphs, setstr(10)=linefeed

```

```

% text box
eTextBoxFeedLine=0; % mm 0=auto else fix linefeed
eTextBoxFeedPara=0; % mm space between paragraphs
eTextBoxSpaceNorth=0; % mm space between text and the north border of box
eTextBoxSpaceSouth=0; % mm space between text and the south border of box
eTextBoxSpaceWest=0; % mm space between text and the north border of box
eTextBoxSpaceEast=0; % mm space between text and the south border of box

% contour
eContourLineWidth=0.2; % mm
eContourLineColor=[0 0 0]; % [r g b] [0 0 0]=black [1 1 1]=white
eContourLineDash=0; % mm 0=solid line >0=dash length
eContourScale=[0 0 0]; % [start step end]
eContourValueVisible=0; % 0=off 1=on
eContourValueFormat=-1; % n digits after decimal point,-1=auto
eContourValueFont=5; % font number 5=Helvetica
eContourValueFontSize=2; % mm
eContourValueDistance=2+eContourLineWidth/2; % mm
eContourLevelsMaxN=10; % max. number of isolevels if autoscaling on

% table
eTabBackgroundColor=[-1 0 0]; % [r g b] if r<0 then transparent
eTabFrameVisible=1; % 0=off 1=on
eTabFrameLineWidth=eLineWidth; % mm
eTabFrameColor=[0 0 0]; % [r g b] [0 0 0]=black [1 1 1]=white
eTabFrameDash=0; % mm 0=solid line >0=dash length
eTabXLineVisible=1; % 0=off 1=on
eTabXLineWidth=eLineWidth; % mm
eTabXLineColor=[0 0 0]; % [r g b] [0 0 0]=black [1 1 1]=white
eTabXLineDash=0; % mm 0=solid line >0=dash length
eTabYLineVisible=1; % 0=off 1=on
eTabYLineWidth=eLineWidth; % mm
eTabYLineColor=[0 0 0]; % [r g b] [0 0 0]=black [1 1 1]=white
eTabYLineDash=0; % mm 0=solid line >0=dash length

```


Chapter 3

User Functions

3.1 Basic Functions

3.1.1 ebbox.m

NAME

ebbox - get and set bounding box

SYNOPSIS

currentBBox=ebbox([newBBox[,epsFileName]])

PARAMETER(S)

currentBBox	current Bounding Box of eps-file
newBBox	default: -1 = read old box coordinates of file if a scalar then newBBox is min. distance (in eUserUnits) between objects and then new calculated bounding box if a vector 1x4 then newBBox are the new bounding box coordinates [x0 y0 x1 y1] in 1/72 inchs
epsFileName	name of eps-file default: 'eFileName'

GLOBAL PARAMETER(S)

eFileName

3.1.2 ebitmap.m

NAME

ebitmap - transform the current eps-file to bitmap-file

SYNOPSIS

mapFileName=ebitmap([bitmapType[,resolution[,mapFileName[,epsFileName]]]])

PARAMETER(S)

bitmapType	bitmap-type default: 0
------------	---------------------------

```

        0 = PNG -format
        1 = JPEG-format
        2 = TIFF-format
        3 = PPM-format
        4 = PCX-format
        5 = PDF-format
resolution    in dpi,resolution of bitmap-file
               if scalar then resolution x and y direction are equal
               default: 200 (dpi)
               if [x y] vector then resolution of x and y direction
               if [x y q] vector then resolution and quality of JPEG
                   q=100 for no lost of quality
                   q=75 standard compression
mapFileName   name of bitmap-file
               default: 'eFileName.typeSuffix'
epsFileName   name of eps-file
               default: 'eFileName'
GLOBAL PARAMETER(S)
eFileName
eGhostscript

```

3.1.3 eclose.m

NAME

eclose - finish plot(s) and close EPS-file

SYNOPSIS

```
eclose ([nCopies[,message]])
```

PARAMETER(S)

```

nCopies       number of hardcopies, for printing 1 or more copies
               default: nCopies=1, print one copy of current page
               if nCopies=0 then 'showpage' will not append
message       switch for 'file written' message
               if message=1 then write message (default)
               else no message

```

GLOBAL PARAMETER(S)

```

eWinFrameVisible
eWinTimeStampVisible

```

3.1.4 eopen.m

NAME

eopen - open EPS-file, define size of page, size of window and call 'einit' to initialize the global parameter

SYNOPSIS

```
eopen([ epsFileName[,pageOrientation[,winWidth,winHeight
[,winShift[,xScaleFac,yScaleFac[,pageWidth,pageHeight
[,pageReflection]]]]]])
```

PARAMETER(S)

epsFileName	name of eps-file (default name is defined as eFileName)
pageOrientation	page orientation, 0=portrait 1=landscape 2=upside-down 3=seaside
winWidth	width of window(=eps bounding-box)
winHeight	height of window(=eps bounding-box)
winShift	shift-vector of window, [xOffset yOffset] ,shift of window center on page, default vector is [0 0]=middle of page
xScaleFac	scale factor 1= no resize
yScaleFac	scale factor 1= no resize
pageWidth	width of page
pageHeight	height of page
pageReflection	refection 1=on 0=off

GLOBAL PARAMETER(S)

```
eFileName
ePageWidth
ePageHeight
ePageScaleFac
ePageOrientation
ePageReflection
eUserUnit
eWinWidth
eWinHeight
eFonts
```

3.1.5 evview.m

NAME

```
evview - start ghostview to show eps-file
```

SYNOPSIS

```
evview([epsFileName])
```

PARAMETER(S)

epsFileName	name of eps-file default: string of global parameter 'eFileName'
-------------	---

GLOBAL PARAMETER(S)

```
eFileName
eGhostview
```

3.2 Axes Functions

3.2.1 eaxes.m

NAME

eaxes - draw scaled axes around plot area

SYNOPSIS

eaxes ([xAxisSouthScale,yAxisWestScale[,xAxisNorthScale,yAxisEastScale]])

PARAMETER(S)

xAxisSouthScale scale vector of south axis [start step end]
 yAxisWestScale scale vector of west axis [start step end]
 xAxisNorthScale scale vector of north axis [start step end]
 yAxisEastScale scale vector of east axis [start step end]

special cases of scale vectors are:

if start=0 and end=0 then autorange=on

if step=0 then autoscale=on

(default scale vector=[0 0 0])

GLOBAL PARAMETER(S)

ePlotAreaXValueStart
 ePlotAreaXValueEnd
 ePlotAreaYValueStart
 ePlotAreaYValueEnd
 ePlotAreaPos
 ePlotAreaWidth
 ePlotAreaHeight
 eAxesValueFontSize
 eAxesValueSpace
 eAxesColor
 eAxesLineWidth
 eAxesTicShortLength
 eAxesTicLongLength
 eAxesTicLongMaxN
 eAxesCrossOrigin
 eAxesValueSpace
 eAxesLabelFontSize
 eAxesLabelTextFont
 eXAxis(South|West|East|North)Scale
 eXAxis(South|West|East|North)ScaleType
 eXAxis(South|West|East|North)Visible
 eXAxis(South|West|East|North)ValueFormat
 eXAxis(South|West|East|North)ValueVisible
 eXAxis(South|West|East|North)LabelText
 eXAxis(South|West|East|North)LabelDistance

3.2.2 eaxespol.m**NAME**

eaxespol - draw scaled axes and arc around polar plot area

SYNOPSIS

```
eaxespol([axisRadiusScale,axisAngleScale])
```

PARAMETER(S)

axisRadiusScale scale vector of radius axis [start step end]

axisAngleScale scale vector of angle circle [start step end]

special cases of scale vectors are:

if start=0 and end=0 then autorange=on

if step=0 then autoscale=on

GLOBAL PARAMETER(S)

ePolarPlotAreaCenterPos

ePolarPlotAreaRadMin

ePolarPlotAreaRadMax

ePolarPlotAreaAngStart

ePolarPlotAreaAngEnd

ePolarPlotAreaValStart

ePolarPlotAreaValEnd

ePolarAxisRadScale

ePolarAxisRadVisible

ePolarAxisRadValueFormat

ePolarAxisRadValueVisible

ePolarAxisRadPos

ePolarAxisAngScale

ePolarAxisAngVisible

ePolarAxisAngValueFormat

ePolarAxisAngValueVisible

ePolarAxisAngValueAngle

eAxesValueFontSize

eAxesColor

eAxesLineWidth

eAxesTicShortLength

eAxesTicLongLength

eAxesTicLongMaxN

eAxesValueSpace

3.2.3 eaxis.m**NAME**

eaxis - draw scaled axis

SYNOPSIS

```
valueTextPos=eaxis(xPos,yPos,length,type,scale[,angle[,color[,offset]]])
```

PARAMETER(S)

xPos x-value of start position of axis
 yPos y-value of start position of axis
 length length of axis
 type orientation 'w'=west, 'e'=east, 's'=south, 'n'=north
 scale vector of scaling, [startValue stepValue endValue]
 angle angle to rotate axis
 color color of axis
 offset offset of position
 valueTextPos positions of values

GLOBAL PARAMETER(S)

eAxesColor
 eAxesValueFontSize
 eAxesValueSpace
 eAxesLineWidth
 eAxesTicShortLength
 eAxesTicLongLength
 eAxesTicLongMaxN
 eXAxisSouthValueFormat
 eYAxisWestValueFormat
 eXAxisNorthValueFormat
 eYAxisEastValueFormat
 eXAxisSouthValueVisible
 eYAxisWestValueVisible
 eXAxisNorthValueVisible
 eYAxisEastValueVisible
 eXAxisSouthValuePos
 eYAxisWestValuePos
 eXAxisNorthValuePos
 eYAxisEastValuePos
 eXAxisSouthScaleType
 eYAxisWestScaleType
 eXAxisNorthScaleType
 eYAxisEastScaleType

3.3 Grid Functions

3.3.1 egrid.m

NAME

egrid - draw grid

SYNOPSIS

egrid([xAxisSouthScale,yAxisWestScale[,xAxisNorthScale,yAxisEastScale]])

PARAMETER(S)

xAxisSouthScale scale vector of south axis [start step end]
 yAxisWestScale scale vector of west axis [start step end]
 xAxisNorthScale scale vector of north axis [start step end]
 yAxisEastScale scale vector of east axis [start step end]

special cases of scale vectors are:

if start=0 and end=0 then autorange=on

if step=0 then autoscale=on

(default scale vector=[0 0 0])

GLOBAL PARAMETER(S)

ePlotAreaPos
 ePlotAreaWidth
 ePlotAreaHeight
 eAxesTicLongMaxN
 eXAxisSouthScale
 eXAxisSouthScaleType
 eYAxisWestScale
 eYAxisWestScaleType
 eXAxisNorthScale
 eYAxisEastScale
 eXGridVisible
 eYGridVisible
 eXGridLineWidth
 eXGridColor
 eXGridDash

3.3.2 egridpol.m

NAME

egridpol - draw polar grid

SYNOPSIS

egridpol ([axisRadiusScale,axisAngleScale])

PARAMETER(S)

axisRadiusScale scale vector of radius axis [start step end]
 axisAngleScale scale vector of angle circle [start step end]

special cases of scale vectors are:

if start=0 and end=0 then autorange=on

if step=0 then autoscale=on

GLOBAL PARAMETER(S)

ePolarPlotAreaValStart
 ePolarPlotAreaValEnd
 ePolarPlotAreaCenterPos
 ePolarPlotAreaRadMin

```

ePolarPlotAreaRadMax
ePolarPlotAreaAngStart
ePolarPlotAreaAngEnd
ePolarAxisRadScale
ePolarAxisAngScale
ePolarRadiusGridVisible
ePolarRadiusGridLineWidth
ePolarRadiusGridColor
ePolarRadiusGridDash
ePolarAngleGridVisible
ePolarAngleGridLineWidth
ePolarAngleGridColor
ePolarAngleGridDash
eAxesTicLongMaxN

```

3.4 Plot Functions

3.4.1 ebar.m

NAME

ebar - get coordinates for bar-plotting

SYNOPSIS

```
[xb yb]=ebar(y[,barWidth[,barNumber,clusterSize,[,x]])
```

PARAMETER(S)

y	vector of y-data
barWidth	x-size of bars if barWidth=0 then autosize default: barWidth=0
number	number within the cluster
clusterSize	total number of bars in one cluster
x	vector of x-data
xb	vector of x-coordinates
yb	vector of y-coordinates

3.4.2 econtour.m

NAME

econtour - draw a contour plot of matrix

SYNOPSIS

```
econtour(matrix[,scale[,dash[,colorMap]]])
```


PARAMETER(S)

matrix matrix for contour plot
 scale vector of scaling [start step end]
 dash if dash=0 then draw solid lines
 else value of dash is the distance of dashes
 colorMap colors for different iso-lines

GLOBAL PARAMETER(S)

ePlotAreaWidth
 ePlotAreaHeight
 ePlotAreaPos
 eContourLineColor
 eContourLineDash
 eContourScale
 eContourLevelsMaxN
 eContourValueFormat
 eContourLineWidth
 eContourValueVisible
 eContourValueDistance
 eContourValueFont
 eContourValueFontSize
 eYAxisWestScale
 eXAxisSouthScale

valueForm=sprintf('1.%.df',vForm);

3.4.3 eerrbar.m

NAME

eerrbar - get coordinates-matrix for errorbar-plotting

SYNOPSIS

[xeb yeb]=eerrbar(x,y,error[,barWidth])

PARAMETER(S)

x vector of x-data
 y vector of y-data
 barWidth x-size of bars
 default: autosize

 xeb matrix of x errorbar-coordinates
 yeb matrix of y errorbar-coordinates

3.4.4 epie.m

NAME

epie - draw a pie chart

SYNOPSIS

```
angles=epie([value[,valueText[,legendText[,dash[,offset[,color]]]]]])
```

PARAMETER(S)

```
value      value of pie slice
valueText  text of value , if empty string then no text at pie slice
legendText text of legend, if empty string then no legend
dash       border type,0=solid line,>0=dash length,
           <0=fill slice with color
offset     radial offset of pieslice, default=0
color      color of pie, vector [r g b]
angles     n x 2 matrix of pie slice angles, if epie without parameter
           [pieSlice1StartAngle pieSlice1SizeAngle;
            pieSlice2StartAngle ...
```

GLOBAL PARAMETER(S)

```
ePolarPlotAreaCenterPos
ePolarPlotAreaRadMax
ePolarPlotAreaValStart
ePolarPlotAreaValEnd
ePolarPlotAreaAngStart;
ePolarPlotAreaAngEnd
ePolarPlotAreaRadMax
ePolarPlotAreaRadMin
ePlotLegendTextFont
ePlotLegendFontSize
ePlotLineDash;
ePlotLineDash;
eAxesTicLongLength
eAxesValueSpace
eAxesValueFontSize
eAxesLineWidth
eAxesColor
ePolarAxisRadScale
ePolarAxisAngScale
```

3.4.5 eplot.m

NAME

```
eplot - make linear plot
```

SYNOPSIS

```
eplot ([xData,[yData,[legendText,[dash,[color[,width]]]]]])
```

PARAMETER(S)

```
xData      vector of x-data
           or matrix(2xn) of x0,x1-data to plot lines
```

yData	vector of y-data or matrix(2xn) of y0,y1-data to plot lines
legendText	text of legend, if empty string then no legend
dash	if a scalar =0 solid plot line, >0 dash length <0 fill plot line with color default: dash=eLineDash if a string then dash is a name of symbol if a matrix and color=-1 dash is the image of plot and filled with RGB values (value=R*2 ¹⁶ +G*2 ⁸ +B and R,G,B are integer of 0:255) if a matrix and color is a colormap dash is the image of plot and filled with indices of colormap if a string dash is filename of a JPEG-file
color	if dash>=0 vector of plot color ([r g b]) if dash<0 vector of background color if dash a matrix then colormap of image or -1 default: dash=eLineColor
width	width of plot line

Important: eplot without parameters closes the current plot explicit.
it's useful for several plot on one page

GLOBAL PARAMETER(S)

```
ePlotAreaPos
ePlotAreaWidth
ePlotAreaHeight
eXAxisSouthScale
eYAxisWestScale
ePlotAreaXValueStart
ePlotAreaXValueEnd
ePlotAreaYValueStart
ePlotAreaYValueEnd
ePlotLineInterpolation
ePlotLineWidth
ePlotLineColor;
ePlotLineDash;
ePlotLegendPos;
ePlotLegendTextFont
ePlotLegendFontSize
ePlotLegendDistance
eAxesColor
```

3.4.6 epolar.m

NAME

epolar - make polar plot

SYNOPSIS

```
epolar ([xData,[yData,[legendText,[dash,[color[,width]]]]]])
```

PARAMETER(S)

xData	vector of x-data
yData	vector of y-data
legendText	text of legend, if empty string then no legend
dash	0=solid line,>0=dash length, <0=fill line,string=name of symbol
color	color of plot, vecor [r g b]
width	width of plot

GLOBAL PARAMETER(S)

```
ePolarAxisRadScale
ePolarAxisAngScale
ePolarPlotAreaCenterPos
ePolarPlotAreaRadMax
ePolarPlotAreaValStart
ePolarPlotAreaValEnd
ePolarPlotAreaAngStart;
ePolarPlotAreaAngEnd
ePolarPlotAreaRadMax
ePolarPlotAreaRadMin
ePlotLegendPos
ePlotLegendTextFont
ePlotLegendFontSize
ePlotLegendDistance;
ePlotLineWidth
ePlotLineColor;
ePlotLineDash;
eAxesColor;
```

3.5 Images Functions

3.5.1 ebright.m

NAME

ebright - change brightness of colormap

SYNOPSIS

```
newColormap=ebright(colormap,brightness[,colorChannel])
```

PARAMETER(S)

colormap	color table
brightness	+/- brighness in per cent
newColormap	changed color table

colorChannel vector of Channel; 1=red, 2=green, 3=blue
e.g. [1 3] = Channel red and blue
default=[1 2 3]

3.5.2 econtra.m

NAME

econtra - change contra of colormap

SYNOPSIS

newColormap=econtra(colormap,contrast[,colorChannel])

PARAMETER(S)

colormap color table
contrast -200 to 200 per cent
newColormap changed color table
colorChannel vector of Channel; 1=red, 2=green, 3=blue
e.g. [1 3] = Channel red and blue
default=[1 2 3]

3.5.3 eidx2rgb.m

NAME

eidx2rgb - converts index-matrix to RGB-matrix

SYNOPSIS

matrix=eidx2rgb(image,colormap)

PARAMETER(S)

image index-matrix
colormap color table
matrix RBG-matrix

3.5.4 eimage.m

NAME

eimage - draw image of a matrix

SYNOPSIS

eimage(matrix[,colorMap])

PARAMETER(S)

matrix rows x cols matrix for image
if colorMap=-1 then

```

        image is filled with RGB values
        (value=R*2^16+G*2^8+B and R,G,B are integer of 0:255)
    else
        matrix is filled with indices of colormap
    or a string of filename of a JPEG-file
colorMap    define own colormap
            default:colorMap=ecolors(eImageDefaultColorMap)

```

GLOBAL PARAMETER(S)

```

eImageDefaultColorMap
ePlotAreaPos
ePlotAreaWidth
ePlotAreaHeight
eImageFrameVisible
eAxesLineWidth

```

3.5.5 eimagesc.m

NAME

eimagesc - draw scaled image of a matrix

SYNOPSIS

```
x=eimagesc(matrix[,colorMap[,legendOrientation[,legendScale]]])
```

PARAMETER(S)

x	transformed output matrix with values of color numbers
matrix	matrix for image
colorMap	define own colormap
legendOrientation	side of the image where the legend appears character 's'(south), 'n'(north), 'w'(west) or 'e'(east) (default orientation is south)
legendScale	scale vector of legend [start step end] special cases of scale vector are: if start=0 and end=0 then autorange=on if step=0 then autoscale=on (default scale vector=[0 0 0])

GLOBAL PARAMETER(S)

```

ePlotAreaPos
ePlotAreaWidth
ePlotAreaHeight
eImageDefaultColorMap
eImageLegendScale
eYAxisWestScale
eXAxisSouthScale

```

3.5.6 eimg2txt.m**NAME**

eimg2txt - converts image to ascii-image

SYNOPSIS

```
txt=eimg2txt(image,colormap)
```

PARAMETER(S)

txt	string of image
image	index-matrix
colormap	color table

3.5.7 eimgmask.m**NAME**

eimgmask - create few masks for image processing

SYNOPSIS

```
imageMask=eimgmask(nRows,nCols[,maskType,[maskPara]])
```

PARAMETER(S)

nRows	number of rows
nCols	number of columns
maskType	type of mask 1: rotation mask, default mask 2: linear mask 3: random mask
maskPara	if maskType==1 then it's radius of maximum, default=1 if maskType==2 then it's direction in deg ,default=-1 <0: maximum in center [0,45) : from N(orth to S(outh) [45,90) : from NE to SW [90,135) : from E to W [135,180): from SE to NW [180,225): from S to N [225,270): from SW to NE [270,315): from W to E [315,360): from NW to SE >=360 : minimum in center if maskType==3 then it's maximum of values, default=1
imageMask	matrix of nRows x nCols with values between 0 and 1

3.5.8 eimgmix.m

NAME

eimgmix - mix two images

SYNOPSIS

```
mixedImage=eimgmix(image1,image2[,fraction[,offset]])
```

PARAMETER(S)

image1 first RGB-image-matrix
 image2 second RGB-image-matrix
 fraction scalar, mixing fraction of image2, ($0 \leq \text{fraction} \leq 1$)
 or vector 1x3, mixing fractions of RGB-Channels
 or matrix with size of image2, mixing fraction for each pixel
 default: fraction=0.5
 offset vector [x y], position of NW-Corner of image2, default [0 0]
 mixedImage mixed RGB-image-matrix

3.5.9 eimgread.m

NAME

eimgread - read image-file

SYNOPSIS

```
[image,colormap]=eimgread(imageFileName)
```

PARAMETER(S)

imageFileName name of JPEG- or PPM-file
 image image matrix
 if colormap used then
 image is filled with indices of colormap
 else
 image is filled with RGB values
 ($\text{value} = R \cdot 2^{16} + G \cdot 2^8 + B$ and R,G,B are integer of 0:255)
 that's a very fast way
 colormap color table

3.5.10 eimgrot.m

NAME

eimgrot - rotate image

SYNOPSIS

```
matrix=eimgrot(image,rotation)
```

PARAMETER(S)

image	index-matrix
rotation	rotation in deg, 90 180 or 270
matrix	RBG-matrix

3.5.11 eimgview.m

NAME

eimgview - create and view eps-file of an image

SYNOPSIS

eimgview(matrix[,colorMap[,epsFileName]])

PARAMETER(S)

matrix	matrix for image if colorMap=-1 then matrix is filled with RGB values (value= $R*2^{16}+G*2^8+B$ and R,G,B are integer of 0:255) else matrix is filled with indices of colorMap or a string of filename of a JPEG-file
colorMap	own colormap default:colorMap=ecolors(eImageDefaultColorMap)
epsFileName	default=eFileName

3.5.12 eimgwrit.m

NAME

eimgwrit - write image-file

SYNOPSIS

eimgwrit(imageFileName,image,colormap,quality)

PARAMETER(S)

imageFileName	name of image-file possible are: png,jpg,tif,ppm,pcx,txt
image	matrix for image if colormap=-1 then image is filled with RGB values (value= $R*2^{16}+G*2^8+B$ and R,G,B are integer of 0:255) else matrix is filled with indices of colormap
colormap	color table
quality	quality of JPEG-files, default=75 (%)

3.5.13 eimgzoom.m

NAME

eimgzoom - resize image

SYNOPSIS

```
[newimage newcolormap]=eimgzoom(image,colormap,scale)
```

PARAMETER(S)

newimage new image matrix
 newimage new color table
 image old image matrix
 colormap color table
 scale scale factor (1=no resize, 2=double size, 0.5=half size)

3.5.14 ejpgread.m

NAME

ejpgread - read JPEG-file

SYNOPSIS

```
[image,head]=ejpgread(jpgFileName)
```

PARAMETER(S)

imageFileName name of JPEG-file e.g. 'photo.jpg'
 image whole JPEG-file in a vector of uchar
 head 1 x 4 vector, [sizeOfJpegFile rowsOfImage colsOfImage rgb]
 rgb=1 if color image, rgb=0 if black and white image

3.5.15 epolari.m

NAME

epolari - draw polar image of a matrix

SYNOPSIS

```
epolari(matrix[,colorMap])
```

PARAMETER(S)

matrix matrix for image
 if colorMap=-1 then
 image is filled with RGB values
 (value= $R*2^{16}+G*2^8+B$ and R,G,B are integer of 0:255)
 else
 matrix is filled with indices of colormap
 colorMap define own colormap
 default:colorMap=ecolors(eImageDefaultColorMap)

GLOBAL PARAMETER(S)

```
eImageDefaultColorMap
ePolarPlotAreaCenterPos
ePolarPlotAreaRadMax
ePolarPlotAreaAngEnd
ePolarPlotAreaAngStart
ePolarPlotAreaRadMin
ePolarPlotAreaRadMax
```

3.5.16 epolaris.m

NAME

```
epolaris - draw scaled polar image of a matrix
```

SYNOPSIS

```
x=epolaris(matrix[,colorMap[,legendOrientation[,legendScale]]])
```

PARAMETER(S)

x	transformed output matrix with values of color numbers
matrix	matrix for image
colorMap	define own colormap
legendOrientation	side of the image where the legend appears character 's' (south), 'n' (north), 'w' (west) or 'e' (east) (default orientation is south)
legendScale	scale vector of legend [start step end] special cases of scale vector are: if start=0 and end=0 then autorange=on if step=0 then autoscale=on (default scale vector=[0 0 0])

GLOBAL PARAMETER(S)

```
eImageDefaultColorMap
eImageLegendScale
ePolarAxisRadScale
ePolarAxisAngScale
ePolarPlotAreaCenterPos
ePolarPlotAreaAngStart
ePolarPlotAreaAngEnd
ePolarPlotAreaRadMax
```

3.5.17 ergb2idx.m

NAME

```
ergb2idx - converts RGB-matrix to index-matrix
```

SYNOPSIS

```
[image,colormap]=ergb2idx(matrix)
```

PARAMETER(S)

matrix	RBG-matrix
image	index-matrix
colormap	color table

3.5.18 ergbsplitt.m

NAME

ergbsplitt - splitt RGB-matrix to red, green and blue matrix

SYNOPSIS

[red,green,blue]=ergbsplitt(matrix)

PARAMETER(S)

matrix	RBG-matrix
red	red color part of matrix,min. value=0,max value=1
green	green color part of matrix,min. value=0,max value=1
blue	blue color part of matrix,min. value=0,max value=1

3.5.19 eshadoi.m

NAME

eshadoi - draw shadow image of a matrix

SYNOPSIS

[x,colorMapNew]=eshadoi(matrix[,colorMap])

PARAMETER(S)

matrix	matrix for image each value of the matrix is a row index of the colormap
colorMap	define own colormap

if the next return parameters are used then no output

x	shadow image matrix
colorMapNew	colormap of x

GLOBAL PARAMETER(S)

eImageDefaultColorMap

3.5.20 eshadois.m

NAME

eshadois - draw scaled shadow image of a matrix

SYNOPSIS

```
[x colorMapNew]=eshadois(matrix[,colorMap[,legendOrientation[,legendScale]]])
```

PARAMETER(S)

matrix	matrix for image
colorMap	define own colormap
legendOrientation	side of the image where the legend appears character 's'(south),'n'(north),'w'(west) or 'e'(east) (default orientation is east)
legendScale	scale vector of legend [start step end] special cases of scale vector are: if start=0 and end=0 then autorange=on if step=0 then autoscale=on (default scale vector=[0 0 0])

if the next return parameters are used then no output

x	scaled shadow image matrix
colorMapNew	colormap of x

GLOBAL PARAMETER(S)

ePlotAreaPos
ePlotAreaWidth
ePlotAreaHeight
eImageDefaultColorMap
eImageLegendScale
eYAxisWestScale
eXAxisSouthScale

3.5.21 eshadoix.m

NAME

eshadoix - mix a shadow image with a cover image

SYNOPSIS

```
[x colorMapNew]=eshadoix(matrix,coverImg,colorMap)
```

PARAMETER(S)

matrix	matrix to calculate the shadow image
coverImg	matrix for cover image each value of this matrix is a row index of the colormap
colorMap	colormap of coverImg

if the next return parameters are used then no output

x	mix shadow image matrix
colorMapNew	colormap of x

3.6 Line Functions

3.6.1 ellipse.m

NAME

ellipse - draw ellipse

SYNOPSIS

```
ellipse(xPos,yPos,width,height[,lineWidth[,dash[,color
[,rotation]]]])
```

PARAMETER(S)

xPos	x-Position of center of ellipse
yPos	y-Position of center of ellipse
width	width of ellipse
height	height of ellipse
lineWidth	linewidth of ellipse
	default: lineWidth=eLineWidth
dash	if a scalar
	=0 solid ellipse,
	>0 dash length
	<0 fill ellipse with color
	default: dash=eLineDash
	if a matrix and color=-1
	dash is the image of ellipse
	and filled with RGB values
	(value=R*2 ¹⁶ +G*2 ⁸ +B and R,G,B are integer of 0:255)
	if a matrix and color is a colormap
	dash is the image of ellipse
	and filled with indices of colormap
	if a string dash is filename of a JPEG-file
color	if dash>=0 vector of ellipse color ([r g b])
	if dash<0 vector of background color
	if dash a matrix then colormap of image or -1
	default: dash=eLineColor
rotation	rotation of ellipse (in deg)

GLOBAL PARAMETER(S)

```
eLineWidth
eLineDash
eLineColor
```

3.6.2 eframe.m

NAME

eframe - draw frame

SYNOPSIS

```
eframe(xPos,yPos,width,height[,lineWidth[,dash[,color
```

```
[,rotation[,cornerRadius]]])
```

PARAMETER(S)

```
xPos      x-Position of sw-corner of frame
yPos      y-Position of sw-corner of frame
width      width of frame
height     height of frame
lineWidth  linewidth of frame
           default: lineWidth=eLineWidth
dash       if a scalar
           =0  solid frame,
           >0  dash length
           <0  fill frame with color
           default: dash=eLineDash
           if a matrix and color=-1
           dash is the image of frame
           and filled with RGB values
           (value=R*2^16+G*2^8+B and R,G,B are integer of 0:255)
           if a matrix and color is a colormap
           dash is the image of frame
           and filled with indices of colormap
           if a string then dash is filename of a JPEG-file
color       if dash>=0 vector of frame color ([r g b])
           if dash<0 vector of background color
           if dash a matrix then colormap of image or -1
           default: dash=eLineColor
rotation    rotation of frame (in deg)
cornerRadius radius of rounded corner
           default: 0=no rounded corner
```

GLOBAL PARAMETER(S)

```
eLineWidth
eLineDash
eLineColor
```

3.6.3 elines.m

NAME

```
elines - draw lines
```

SYNOPSIS

```
elines(xData,yData[,lineWidth[,dash[,color]]])
```

PARAMETER(S)

```
xData      matrix(2xn) of x0,x1-data of lines
yData      matrix(2xn) of y0,y1-data of lines
lineWidth  width of lines
           default: lineWidth=eLineWidth
dash       if dash=0 then draw solid lines
```

else value of dash is the distance of dashes
 default: dash=eLineDash
 color vector of line color ([r g b])
 default: color=eLineColor

GLOBAL PARAMETER(S)

eLineWidth
 eLineDash
 eLineColor

3.6.4 epline.m

NAME

epline - draw polyline

SYNOPSIS

epline(xData,yData[,lineWidth[,dash[,color]]])

PARAMETER(S)

xData vector of x-values of polyline
 yData vector of y-values of polyline
 lineWidth width of polyline
 default: lineWidth=eLineWidth
 dash if a scalar
 =0 solid lines,
 >0 dash length
 <0 fill polylines with color
 default: dash=eLineDash
 if a matrix and color=-1
 dash is the image of polyline
 and filled with RGB values
 (value=R*2¹⁶+G*2⁸+B and R,G,B are integer of 0:255)
 if a matrix and color is a colormap
 dash is the image of polyline
 and filled with indices of colormap
 if a string then dash is filename of a JPEG-file
 color if dash>=0 vector of frame color ([r g b])
 if dash<0 vector of background color
 if dash a matrix then colormap of image or -1
 default: dash=eLineColor

GLOBAL PARAMETER(S)

eLineColor
 eLineDash
 eLineWidth

3.7 Text Functions

3.7.1 eparam.m

NAME

eparam - print parameter text in two columns under plots

SYNOPSIS

eparam(text1,text2[,x,y])

PARAMETER(S)

text1	text of the left column
text2	text of the right column
x	x-coordinate of start position
y	y-coordinate of start position

GLOBAL PARAMETER(S)

eParamPos
eParamFontSize
eParamTextValueDistance
eParamTextFont
eParamValueFont
eParamLineDistance

3.7.2 etext.m

NAME

etext - write text

SYNOPSIS

etext(text[,x[,y[,fontSize[,alignment[,font[,rotation[,color]]]]]]])

PARAMETER(S)

text	text string
x	x of start position if x=0 then the text starts after the last text in the same line
y	y of start position if x=0 then y is a relativ position to the current line
fontSize	scalar size of current font or vector [xSize ySize obliqueAngle(in deg)] of current font
alignment	1=right 0=center -1=left from x-positon, y = line position 2=right 3=center 4=left from x-positon, y = height of text/2
font	font number (definition in einit.m)
rotation	rotation of text (in deg)
color	color of text, [r g b] vector

GLOBAL PARAMETER(S)

eTextColor

```
eTextRotation
eTextFont
eTextAlignment
eTextFontSize
```

3.7.3 etxtbox.m

NAME

```
etxtbox - write text in a box
```

SYNOPSIS

```
etxtbox(text[,x[,y[,boxWidth[,boxHeight[,fontSize[,alignment
[,font[,rotation[,color[,offset]]]]]]]]]])
```

PARAMETER(S)

text	text string
x	x of start position
y	y of start position
boxWidth	width of textbox default=eWinWidth
boxHeight	height of textbox
fontSize	scalar size of current font or vector [xSize ySize obliqueAngle(in deg)] of current font
alignment	1=right 0=center -1=left 2=block
font	font number (definition in einit.m)
rotation	rotation of box (in deg)
color	color of text, [r g b] vector
offset	offset vector [x y] of text, default offset=[0 0]

GLOBAL PARAMETER(S)

```
eTextColor
eTextFont
eTextAlignment
eTextFontSize
eTextLimitWord
eTextLimitPara
eTextBoxFeedLine
eTextBoxFeedPara
```

3.7.4 etxtlpos.m

NAME

```
etxtlpos - get text line positions
```

SYNOPSIS

```
[linePos,nLines]=etxtlpos(text[,lineLimit])
```

PARAMETER(S)

text	sting of text
lineLimit	sting for splitting lines (default=eTextLimitPara)
linePos	nLines x 2 Matrix of start and end positions [line1StartPos line1EndPos; line2StartPos line2EndPos ...
nLines	number of lines

3.7.5 etxtread.m

NAME

etxtread - read text-file

SYNOPSIS

[text,textLength]=etxtread(textFileName)

PARAMETER(S)

textFileName	name of textfile
text	sting of text

3.7.6 etxtwrit.m

NAME

etxtwrit - write string to text-file

SYNOPSIS

etxtwrit(text,textFileName)

PARAMETER(S)

text	sting of text
textFileName	name of textfile

3.8 Symbol Functions

3.8.1 edsymbol.m

NAME

edsymbol - define symbols for plotting

SYNOPSIS

edsymbol(name,symbolFileName
[,scaleX[,scaleY[,moveX[,moveY[,rotation[,color]]]]]])

PARAMETER(S)

name	definition name for new symbol
symbolFileName	pathname of an own postscript symbol definition file

	or filename of a standard psd-file in the "ePath"-directory
	or string of postscript commands
scaleX	scale factor in X-direction
scaleY	scale factor in Y-direction
moveX	offset in X-direction
moveY	offset in X-direction
rotation	rotate symbol (deg)
color	color vector [r g b], color of symbol

3.8.2 equiver.m

NAME

equiver - draw a quiver plot of matrix

SYNOPSIS

equiver(xData,yData,dx,dy[,color[,symbolName]])

PARAMETER(S)

xData	vector or matrix of x-positions of the symbols
yData	vector or matrix of y-positions of the symbols
dx	vector or matrix of x-values to determine the direction and relative magnitude of the symbols
dy	vector or matrix of y-values to determine the direction and relative magnitude of the symbols
color	color of symbols, vector [r g b]
symbolName	symbol name of edsymbol() function default symbol is an arrow

GLOBAL PARAMETER(S)

ePlotAreaPos
ePlotAreaWidth
ePlotAreaHeight
eXAxisSouthScale
eYAxisWestScale
ePlotAreaXValueStart
ePlotAreaXValueEnd
ePlotAreaYValueStart
ePlotAreaYValueEnd

3.8.3 esymbol.m

NAME

esymbol - draw a defined symbol

SYNOPSIS

esymbol(xPos,yPos,symbolName[,scaleX[,scaleY[,rotation]]])

PARAMETER(S)

xPos	x position
yPos	y position
symbolName	name of defined symbol
scaleX	scale factor in x-direction
scaleY	scale factor in y-direction
rotation	rotate symbol (deg)

3.9 Table Functions

3.9.1 etabdef.m

NAME

etabdef - defines a table

SYNOPSIS

```
[colsXW rowsYH]=etabdef(rows,cols[,x,y[,width,height
                        [,colsWidth[,rowsHeight]]])
```

PARAMETER(S)

rows	number of rows
cols	number of columns
x	x-position (sw-corner) of table
y	y-position (sw-corner) of table
width	width of table
height	height of table
colsWidth	vector of relative width of columns ([1 1 1 ... 1]==equal widths)
rowsHeight	vector of rel. height of columns ([1 1 1 ... 1]==equal heights)
colsXW	matrix of x-positions and width of columns, size of matrix is (number of cols) X 2 examples: colsXW(5,1) = x-position of column 5 colsXW(5,2) = width of column 5
rowsYH	matrix of y-positions and height of rows, size of matrix is (number of rows) X 2 examples: cellXW(5,1) = y-position of row 5 cellXW(5,2) = height of row 5

GLOBAL PARAMETER(S)

eTabBackgroundColor

3.9.2 etabgrid.m

NAME

etabgrid - draw lines of table

SYNOPSIS

```
etabgrid(colsXW,rowsYH)
```

PARAMETER(S)

colsXW matrix of x-positions and width of columns,
 size of matrix is (number of cols) X 2
 examples: cellXW(5,1) = x-position of column 5
 cellXW(5,2) = width of column 5

rowsYH matrix of y-positions and height of rows,
 size of matrix is (number of rows) X 2
 examples: cellXW(5,1) = y-position of row 5
 cellXW(5,2) = height of row 5

GLOBAL PARAMETER(S)

3.9.3 etabtext.m

NAME

etabtext - fill cell of table with text

SYNOPSIS

etabtext(colsXW,rowsYH,row,col,text[,alignment
 [,font[,fontSize[,color[,bgColor]]]])

PARAMETER(S)

colsXW matrix of x-positions and width of columns,
 size of matrix is (number of cols) X 2
 examples: cellXW(5,1) = x-position of column 5
 cellXW(5,2) = width of column 5

rowsYH matrix of y-positions and height of rows,
 size of matrix is (number of rows) X 2
 examples: cellXW(5,1) = y-position of row 5
 cellXW(5,2) = height of row 5

row number of row

col number of column

text text of cell

alignment 1=right 0=center -1=left

font font number (definition in einit.m)

fontSize relative fontsize in percent (100=default)

color color of text

bgColor color of background, [r g b] vector, if r<0 then transparent

GLOBAL PARAMETER(S)

eTextColor

eTextFont

3.10 Special Functions

3.10.1 ecdcover.m

NAME

ecdcover - write a cdcover

SYNOPSIS

```
ecdcover(title[,description[,author[,version[,date[,textColor
            [,frontImage[,background[,logo[,content]]]]]]]]))
```

PARAMETER(S)

title	string of title of cd
description	string of description, default=''
author	string of author, default=''
version	string of version, default=''
date	string of time or date, default=''
textColor	color vector [r g b] [0 0 0]=black [1 1 1]=white, default=[0 0 0]
frontImage	filename of frontImage , default=''
background	filename of background, default=''
logo	filename of logo, default=''
content	filename(ascii-file, tabs with '#') of table of contents

3.10.2 ecolorm.m

NAME

ecolorm - get a colormap defined in einit.m

SYNOPSIS

```
colorMap=ecolorm([mapNo [,nColors]])
```

PARAMETER(S)

colorMap	matrix of nColors x 3 Values between 0 and 1
mapNo	map number in the definition matrix eColorMaps (default=0)
nColors	number of colors (default=64)

GLOBAL PARAMETER(S)

eColorMaps

3.10.3 efillmat.m

NAME

efillmat - fill matrix with interpolated values by given xyz-samples

SYNOPSIS

```
matrix=efillmat(xData,yData,zData,dX,dY)
```

PARAMETER(S)

xData	vector of x-coordinates
yData	vector of y-coordinates
zData	vector of z-values
dx	pixel distance of x-direction
dy	pixel distance of y-direction
matrix	interpolated matrix

3.10.4 egradient.m

NAME

egradient - get numerical partial derivatives of matrix

SYNOPSIS

[px py]=egradient(z[,dx[,dy]])

PARAMETER(S)

px	px=dz/dx
py	py=dz/dy
z	z-matrix
dx	delta x
dy	delta y

3.10.5 einflate.m

NAME

einflate - inflate columns of matrix

SYNOPSIS

newMatrix=einflate(matrix[,exponent])

PARAMETER(S)

matrix	real matrix
exponent	exponent of factor 10^{exponent} default=2
newMatrix	normalized new matrix

3.10.6 einseps.m

NAME

einseps - insert eps-file

SYNOPSIS


```
einseps(xPos,yPos,epsFileName,[,scaleX[,scaleY[,rotation]]])
```

PARAMETER(S)

```
xPos      x position
yPos      y position
epsFileName  name of eps-file
scaleX    scale factor in x-direction
scaleY    scale factor in y-direction
```

3.10.7 eisoline.m

NAME

```
eisoline - get isolines of a matrix
```

SYNOPSIS

```
lines=eisoline(matrix,isoValue)
```

PARAMETER(S)

```
lines      empty matrix or 2n x 2 matrix,
            n=number of lines x [x1 y1;x2 y2]
matrix     matrix of values
isoValue   value of isoline
```

3.10.8 ejpglist.m

NAME

```
ejpglist - generate photoprints of a JPEG-filelist
```

SYNOPSIS

```
ejpglist([listFileName[,maxPhotoSize[,fitPhoto[,outputFileName]]]])
```

PARAMETER(S)

```
listFileName      textfile of JPEG-filenames
                   one name per line
                   default=current directory
maxPhotoSize      vector [width height] of photos
                   default=[90 120] (90mmx120mm)
fitPhoto          switch, 0=off 1=fit photos to maxPhotoSize,default=0
outputFileName    Praefix of eps-outputfile
                   default='photos' ->photos01.jpg,photos02.jpg, ...
```

3.10.9 elineip.m

NAME

```
elineip - linear interpolation of a vector
```

SYNOPSIS

```
yi=elineip(x,y,xi)
```

PARAMETER(S)

x	sample x vector
y	sample y vector
xi	x vector for interpolation
yi	interpolated y vector

3.10.10 emerge.m

NAME

`emerge` - merge matrix

SYNOPSIS

```
matrixNew=emerge(matrixOld,key[,direction])
```

PARAMETER(S)

matrixOld	nxm matrix
key	vector of integer for merging (can be used as key)
direction	1 merge forward, -1 merge backward,default=1

3.10.11 eplo2win.m

NAME

`eplo2win` - transform coordinates , plotarea to window

SYNOPSIS

```
[winX winY]=eplo2win(ploX,plotY)
```

PARAMETER(S)

ploX	x-vector of coordinates of plotarea
ploY	y-vector of coordinates of plotarea
winX	x-vector of coordinates of window
winY	y-vector of coordinates of window

GLOBAL PARAMETER(S)

3.10.12 eshadow.m

NAME

`eshadow` - make shadow image matrix

SYNOPSIS

```
x=eshadow(matrix,nColors,colorMap,lumen,image)
```

PARAMETER(S)

matrix	matrix for image
nColors	number of colors
colorMap	(nColors*nBrightnessLevels) x 3 Matrix
lumen	light direction, [x,y,z] vector
image	cover image
x	shadow image matrix

3.10.13 esubeps.m

NAME

```
esubeps - insert eps-file in a subarea of the window
```

SYNOPSIS

```
esubeps(nRows,nColumns,row,column,epsFileName)
```

PARAMETER(S)

nRows	number of rows of the window
nColumns	number of columns of the windows
row	index of row
column	index of column
epsFileName	name of eps-file

3.10.14 ewinsize.m

NAME

```
ewinsize - get size of Bounding Box of eps-file
```

SYNOPSIS

```
[width,height]=ewinsize([epsFileName])
```

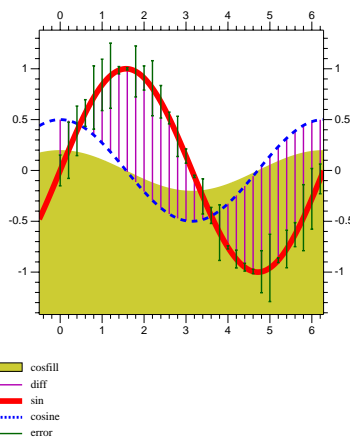
PARAMETER(S)

epsFileName	name of eps-file
	default: current eFileName

Appendix A

A.1 Examples

A.1.1 edemo1.m



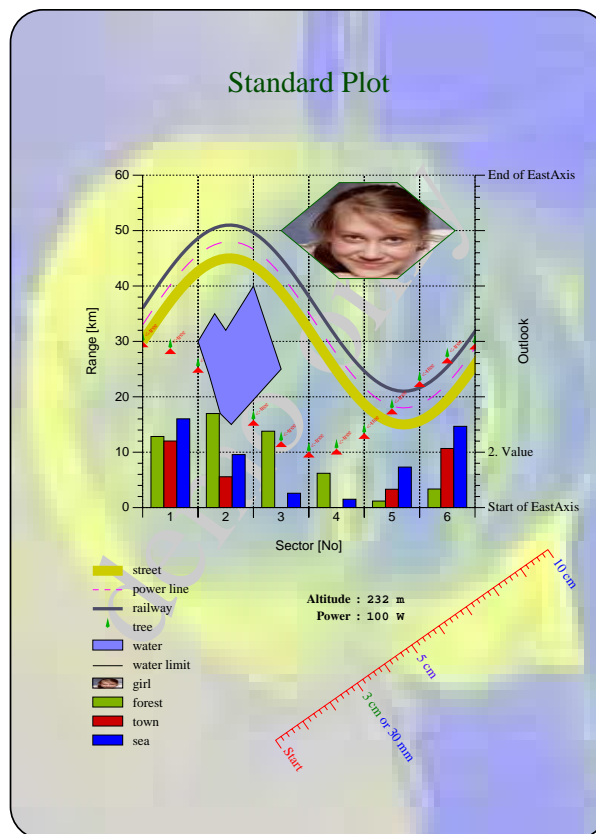
```
x1=[-0.5:0.01:2*pi];  
x2=[0:0.2:2*pi];  
  
eopen('demo1.eps') % open eps-file and write eps-head  
  
% fill area  
eplot(x1,cos(x1)*0.2,'cosfill',-1,[0.8 0.8 0.2])  
  
% plot single lines  
eplot([x2;x2],[cos(x2)*0.5;sin(x2)],'diff',0,[0.7 0 0.7],0.5)  
  
% plot red solid line  
eplot(x1,sin(x1),'sin',0,[1 0 0],2)  
  
% plot blue dash line  
eplot(x1,cos(x1)*0.5,'cosine',2,[0 0 1],1)
```

```

% plot errorbar
error=rand(size(x2))/3;
[xm ym]=eerrbar(x2,sin(x2),error);
eplot(xm,ym,'error',0,[0 0.4 0],0.5);
eclose
newbox=ebbox(5);
if ~exist('noDemoShow')
    eview
end
% close eps-file
% make new boundingbox with 5mm frame
% start ghostview with eps-file

```

A.1.2 edemo2.m



```

% standard plot
eopen('demo2.eps')
eglobpar
eXAxisSouthLabelText='Sector [No]';
eXAxisSouthScaleType=1;
eYAxisWestLabelText='Range [km]';
eYAxisEastLabelText='Outlook';
% open eps-file and write eps-head
% get access to global parameters
% set South Label of XAxis
% set classes scaling
% set West Label of YAxis
% set East Label of YAxis

```

```

eXAxisNorthVisible=0;           % switch North-XAxis off
eYAxisEastValueVisible=0;       % switch East-YAxis Values off
eXGridVisible=1;                % switch x-Grid on
eYGridVisible=1;                % switch y-Grid on

% background
[im cm]=eimgread; % read standard image
cm=econtra(cm,200,[1 2]); % change contrast of image
cm=ebright(cm,60); % change brightness of image
im=eimgrot(im,90); % rotate image
eframe(0,0,eWinWidth,eWinHeight,0,im,cm,0,10); % print background image

% frame
eframe(0,0,eWinWidth,eWinHeight,0.5,0,[0 0 0],0,10); % draw frame around window
etext('demo only',eWinWidth/2,eWinHeight/2,40,0,1,... %print demo text
      atan(eWinHeight/eWinWidth)*180/pi,[0.8 0.8 0.8])
eXAxisSouthScale=[0 0 6]; %set scale x-axis
eYAxisWestScale=[0 10 60]; %set scale y-axis

etitle('Standard Plot',25,9,[0 0.3 0])           % print title

% 1. plot lines
x=0:0.1:2*pi;
eplot(x,sin(x)*15+30,'street',0,[0.8 0.8 0],3) %solid line
eplot(x,sin(x)*15+33,'power line',5,[1 0 1]) % dash plot
eplot(x,sin(x)*15+36,'railway',0,[0.3 0.3 0.4],1) % solid line

% 2. plot symbols
x=0:0.5:2*pi;
edsymbol('spire','spire.psd',0.3,0.3,0,0,90) % define symbol 'spire' with a symbol file
eplot(x,cos(x)*10+20,'tree','spire',[0 0.7 0]) % plot trees
edsymbol('textplot','10 0 moveto (<-tree) show',0.5,0.5,0,0,45) % define symbol 'textplot' with post
text='newpath 0 0 moveto -10 -10 lineto 10 -10 lineto closepath fill';
etxtwrit(text,'myNewSymbol.psd'); % write new symbol file
edsymbol('mysymbol','myNewSymbol.psd',0.5,0.5) % define symbol 'mysymbol'

eplot(x,cos(x)*10+20,'','textplot',[1 0 0]) % plot trees
eplot(x,cos(x)*10+20,'','mysymbol',[1 0 0]) % plot mysymbol

% 3. plot area
lake=[1 30;1.3 35;1.5 32;2 40;2.5 25;1.6 15;1.4 17;1.2 24;1 30];
eplot(lake(:,1),lake(:,2),'water',-1,[0.5 0.5 1]); % filled area,dash<0
eplot(lake(:,1),lake(:,2),'water limit',0,[0 0 0]); % solid line around the area

% 4. plot image
x1=0:pi/3:pi;
x1=x1';
x2=flipud(x1);
girl=[x1+2.5 sin(x1)*10+50;x2+2.5 -sin(x2)*10+50];
im=eimgread;

```

```

epplot(girl(:,1),girl(:,2),'girl',im,-1); % fill area with image
epplot(girl(:,1),girl(:,2),'',0,[0 0.4 0]); % solid line around the image

% 5. plot bars
x=0.5:1:5.5;
[xb yb]=ebar(sin(x)*8+9,0,1,3); % 1. bars
epplot(xb,yb,'forest',-1,[0.5 0.7 0])
epplot(xb,yb,'',0,[0 0 0])
[xb yb]=ebar(cos(x)*8+5,0,2,3); % 2. bars
epplot(xb,yb,'town',-1,[0.8 0 0])
epplot(xb,yb,'',0,[0 0 0])
[xb yb]=ebar(cos(x)*8+9,0,3,3); % 3. bars
epplot(xb,yb,'sea',-1,[0 0 1])
epplot(xb,yb,'',0,[0 0 0])
epplot

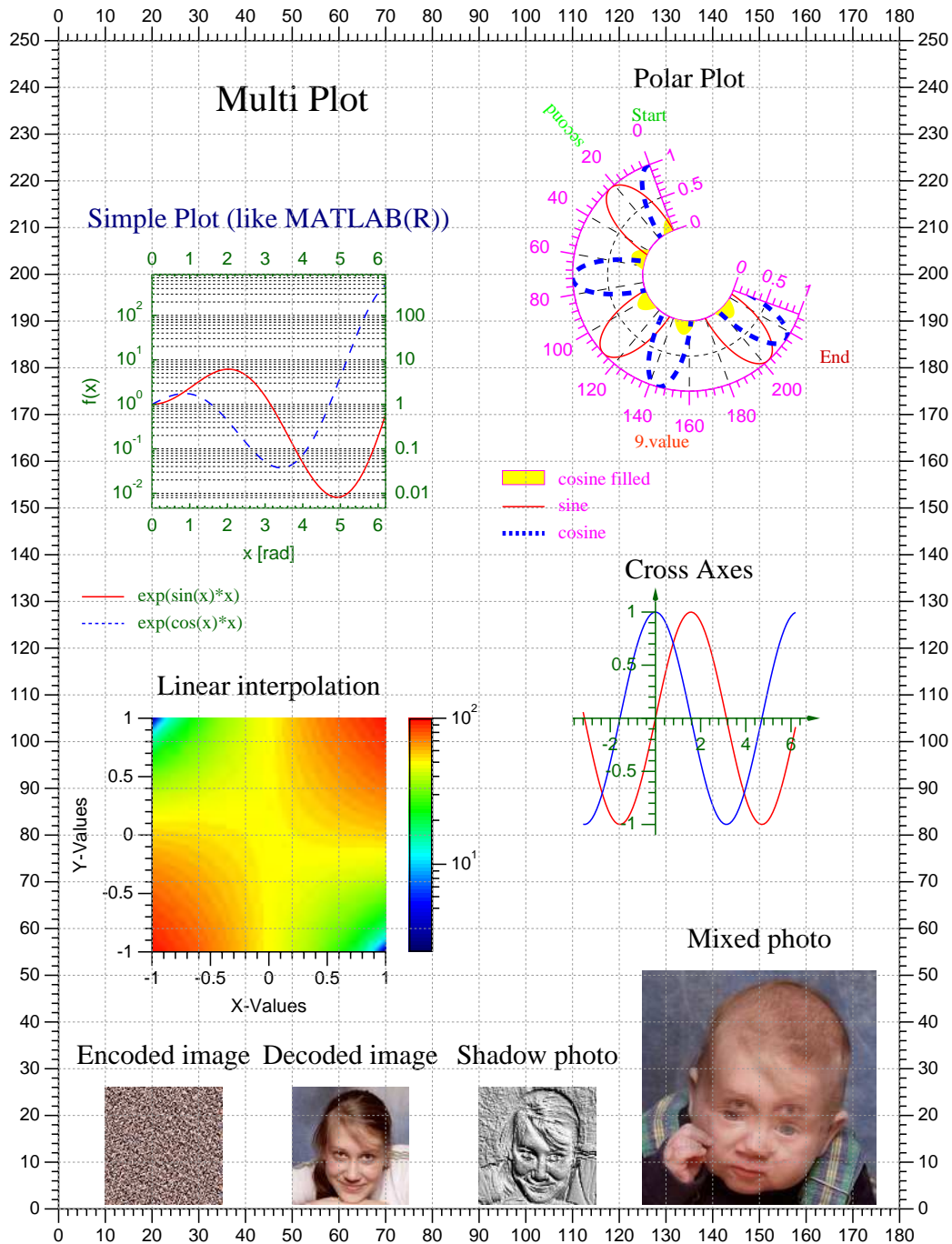
% write parameters
eParamPos=[80,70];
eParamFontSize=5;
eparam('Altitude','232 m')
eparam('Power','100 W')
valuePos=eYAxisEastValuePos;
posNo=1;
etext('Start of EastAxis',valuePos(posNo,1),valuePos(posNo,2),4,4);
posNo=2;
etext('2. Value',valuePos(posNo,1),valuePos(posNo,2),4,4);
posNo=7;
etext('End of EastAxis',valuePos(posNo,1),valuePos(posNo,2),4,4);

% axis
eYAxisEastScaleType=0; % set linear scaling
eYAxisEastValueVisible=0; % no values visible
angle=-55;
eaxis(80,30,100,'e',[0 1 10],angle,[1 0 0]);
valuePos=eYAxisEastValuePos;
posNo=1;
etext('Start',valuePos(posNo,1),valuePos(posNo,2),4,1,1,angle,[1 0 0]);
posNo=4;
etext('3 cm',valuePos(posNo,1),valuePos(posNo,2),4,4,1,angle,[0 0.5 0]);
etext(' or 30 mm',0,0,4,4,1,angle,[0 0 1]);
posNo=6;
etext('5 cm',valuePos(posNo,1),valuePos(posNo,2),4,4,1,angle,[0.3 0 1]);
posNo=11;
etext('10 cm',valuePos(posNo,1),valuePos(posNo,2),4,4,1,angle,[0 0 1]);

eclose % close ps output
if ~exist('noDemoShow')
    eview % start ghostview with eps-file
end

```


A.1.3 edemo3.m



```

eopen('demo3.eps');
eglobpar;                % get access to global parameters
eWinGridVisible=1;
esavpar;                 % save default parameter

```

```

%title
etext('Multi Plot',50,eWinHeight-15,8,0,1)

% set tics like Matlab
eAxesValueSpace=3;
eAxesTicLongLength=-1;
eAxesTicShortLength=-0.5;

% simple plot
eAxesColor=[0 0.4 0];
ePlotAreaPos=[20 150];
ePlotAreaHeight=50;
ePlotAreaWidth=50;
eYAxisWestLabelText='f(x)';
eXAxisSouthLabelText='x [rad]';
etitle('Simple Plot (like MATLAB(R))',10,6,[0 0 0.5]);
eYAxisWestScaleType=2;
eYGridVisible=1;
xData=0:0.1:2*pi;
eYAxisEastValueFormat=2;
eplot(xData,exp(sin(xData).*xData),'exp(sin(x)*x)',0,[1 0 0]);
eplot(xData,exp(cos(xData).*xData),'exp(cos(x)*x)',2,[0 0 1]);
eplot;

% polar plot
erespar; % set default parameter
eAxesColor=[1 0 1];
ePolarPlotAreaCenterPos=[135 200];
ePolarPlotAreaRadMin=10;
ePolarPlotAreaRadMax=25;
ePolarPlotAreaAngStart=110;
ePolarPlotAreaAngEnd=340;
ePolarAxisRadVisible=3;
ePolarAxisRadValueVisible=3;
ePolarAxisRadScale=[0 0.5 1];
ePlotTitleDistance=15;
ePlotTitleText='Polar Plot';
xData=0:0.01:2*pi;
epolar(xData,cos(xData*7)*0.2,'cosine filled',-1,[1 1 0]);
epolar(xData,sin(xData*4),'sine',0,[1 0 0]);
epolar(xData,cos(xData*5),'cosine',2,[0 0 1],1);
epolar;
angles=ePolarAxisAngValueAngle*pi/180;
dis=11;
lPos=[cos(angles) sin(angles)]*(ePolarPlotAreaRadMax+dis);
lPos=[lPos(:,1)+ePolarPlotAreaCenterPos(1)...
      lPos(:,2)+ePolarPlotAreaCenterPos(2)];
i=1;
etext('Start',lPos(i,1),lPos(i,2),4,4,1,0,[0 0.8 0]);
i=2;

```

```

etext('second',lPos(i,1),lPos(i,2),4,4,1,ePolarAxisAngValueAngle(i),[0 1 0]);
i=9;
etext('9.value',lPos(i,1),lPos(i,2),4,2,1,0,[1 0.2 0]);
i=size(lPos,1);
ellipseW=2*(dis-eAxesTicLongLength);
eellipse(lPos(i,1),lPos(i,2),ellipseW,ellipseW,0,-1,[1 1 1]); % cover value
etext('End',lPos(i,1),lPos(i,2),4,3,1,0,[0.8 0 0]);

% cross axes plot
erespar; % set default parameter
eAxesColor=[0 0.4 0];
ePlotAreaPos=[110 80];
ePlotAreaHeight=50;
ePlotAreaWidth=50;
ePlotTitleDistance=5;
ePlotTitleText='Cross Axes';
eAxesCrossOrigin=2; % cross axes with arrows
xData=-3.2:0.1:6.2;
eplot(xData,sin(xData),'',0,[1 0 0]);
eplot(xData,cos(xData),'',0,[0 0 1]);
eplot

% interpolation image
erespar; % set default parameter
ePlotAreaPos=[20 55];
ePlotAreaHeight=50;
ePlotAreaWidth=50;
eYAxisWestLabelText='Y-Values';
eXAxisSouthLabelText='X-Values';
eImageLegendPos=[0 -5];
ePlotTitleDistance=5;
ePlotTitleText='Linear interpolation';
eXAxisNorthVisible=0;
eYAxisEastVisible=0;
matrix=efillmat([1 -1 -1 1],[1 1 -1 -1],[100 1 100 1],0.02,0.02);
matrix=einflate(matrix,0.8);
eImageLegendScaleType=2;
eXAxisSouthScale=[-1 0 1]; %set scale x-axis
eYAxisWestScale=[-1 0 1]; %set scale y-axis
eimagesc(matrix,ecolors(3),'e'); % print scaled image

% mixing photos
erespar; % set default parameter
ePlotAreaPos=[125 1];
ePlotAreaHeight=50;
ePlotAreaWidth=50;
ePlotTitleDistance=5;
ePlotTitleText='Mixed photo';
eImageLegendVisible=0;
img1=eimgread([ePath 'default.jpg']);

```

```

img2=eimgread;
img2=eimgzoom(img2,-1,1.9); % resize image
[img1Height img1Width]=size(img1);
[img2Height img2Width]=size(img2);
imgMask=eimgmask(img2Height,img2Width,1,0.6);
img2Pos=[fix((img1Height-img2Height)/2) fix((img1Width-img2Width)/2)];
photo=eimgmix(img1,img2,imgMask,img2Pos); % mixing
eimage(photo,-1);

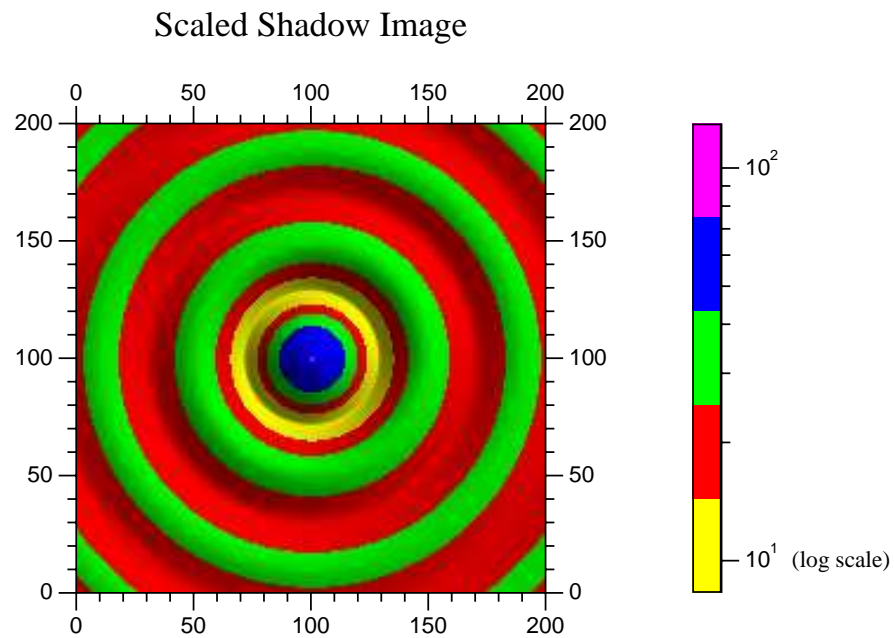
% shadow photo
erespar; % set default parameter
ePlotAreaPos=[90 1];
ePlotAreaHeight=25;
ePlotAreaWidth=25;
ePlotTitleDistance=5;
ePlotTitleText='Shadow photo';
[photo colormap]=eppmread; % read default image
eshadoi(photo); % print shadow image

% encode and decode iamges
ePlotTitleText='Encoded image';
photo=emerge(photo,[47 11]); % merge pixel
ePlotAreaPos=[10 1];
eimage(photo,colormap);
ePlotTitleText='Decoded image';
photo=emerge(photo,[47 11],-1); % merge pixel backwards
ePlotAreaPos=[50 1];
eimage(photo,colormap);

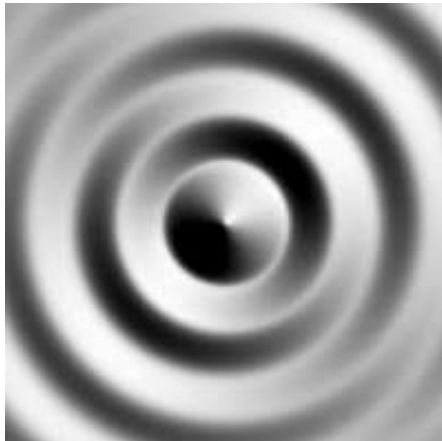
% close eps-file
eclose;
if ~exist('noDemoShow')
    eview % start ghostview with eps-file
end

```

A.1.4 edemo4.m



Shadow Image



Mixed Shadow Image

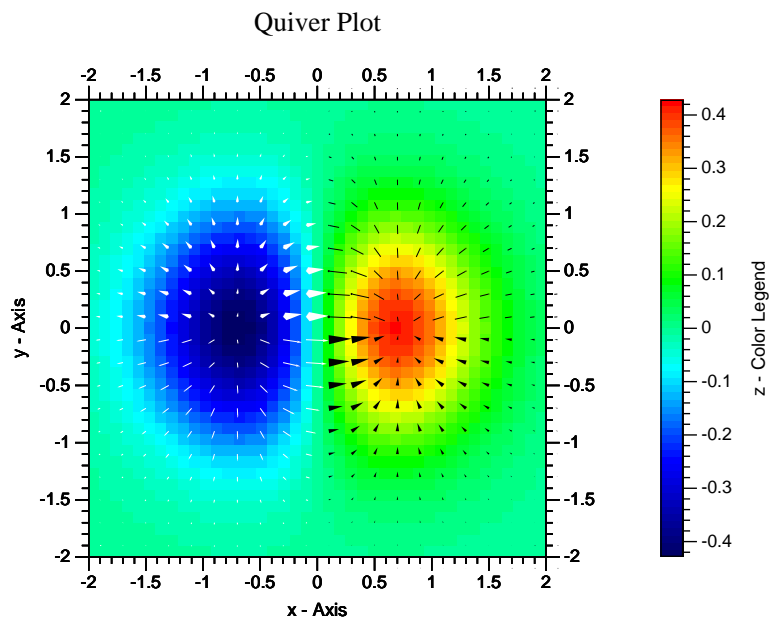
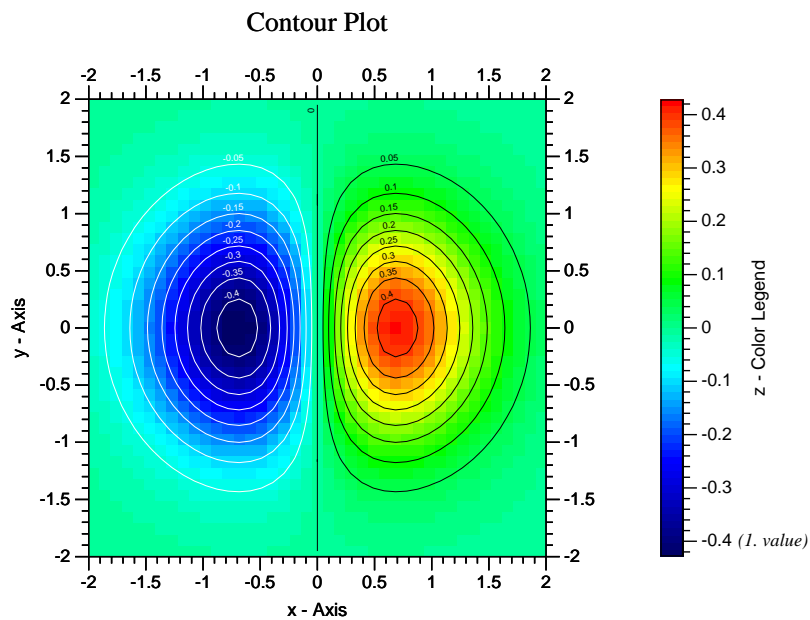


```
eopen('demo4.eps');
eglobpar;

[img cm]=eppmread([ePath 'defMap.ppm']);
n=size(img,1);
```

```
delta=10*pi/(n-1);  
x=-5*pi:delta:5*pi;  
[a b]=meshgrid(x,x);  
R=sqrt(a.^2+b.^2) + eps;  
matrix=100*(sin(R)./R+0.3);  
  
% scaled shadow image  
ePlotAreaPos=[30 140];  
ePlotAreaHeight=80;  
ePlotAreaWidth=80;  
ePlotTitleDistance=15;  
eImageLegendScaleType=2;  
ePlotTitleText='Scaled Shadow Image';  
colorMap=[1 1 0;1 0 0;0 1 0;0 0 1;1 0 1];  
eshadois(matrix,colorMap,'e');  
etext('          (log scale)',eImageLegendValuePos(1,1),...  
      eImageLegendValuePos(1,2),4,4,1);  
  
% shadow image  
ePlotAreaPos=[10 30];  
ePlotAreaHeight=75;  
ePlotAreaWidth=75;  
ePlotTitleDistance=5;  
ePlotTitleText='Shadow Image';  
eshadoi(matrix);  
  
% mixed shadow image  
ePlotAreaPos=[90 30];  
ePlotAreaHeight=75;  
ePlotAreaWidth=75;  
ePlotTitleDistance=5;  
ePlotTitleText='Mixed Shadow Image';  
eshadoix(matrix,img,cm);  
  
eclose;  
newbbox=eboxed(5);  
if ~exist('noDemoShow')  
    eview                                     % start ghostview with eps-file  
end
```

A.1.5 edemo5.m



```

x=-2:.1:2;
y=-2:.1:2;
[X,Y]=meshgrid(x,y);
matrix=X.*exp(-X.^2-Y.^2);

```

```
% contour plot
eopen('demo5a.eps',0,180,140);
eglobpar;
ePlotTitleText='Contour Plot';
ePlotTitleDistance=15;
ePlotAreaPos=[20 20];
eXAxisSouthLabelText='x - Axis';
eYAxisWestLabelText='y - Axis';
eImageLegendLabelText='z - Color Legend';
eContourValueVisible=1;
eaxes([-2 0 2],[-2 0 2]);
eimagesc(matrix,ecolors(3),'e');
etext('      (1. value)',eImageLegendValuePos(1,1),...
      eImageLegendValuePos(1,2),4,4,2);
econtour(matrix,[-0.5 0.05 0.5],0,[1 1 1;0 0 0;0 0 0]);
eclose(1,0);
```

```
% quiver plot
eopen('demo5b.eps',0,180,140);
eglobpar;
ePlotTitleText='Quiver Plot';
ePlotTitleDistance=15;
ePlotAreaPos=[20 20];
eXAxisSouthLabelText='x - Axis';
eYAxisWestLabelText='y - Axis';
eImageLegendLabelText='z - Color Legend';
eaxes([-2 0 2],[-2 0 2]);
eimagesc(matrix,ecolors(3),'e');
```

```
% sw
x=-2.1:.2:-0.1;
y=-2.1:.2:-0.1;
[X,Y]=meshgrid(x,y);
qmatrix=X.*exp(-X.^2-Y.^2);
[dx dy]=egradient(qmatrix,.2,.2);
equiver(X,Y,dx,dy,[1 1 1]);
```

```
% nw
x=-2.1:.2:-0.1;
y=0.1:.2:2.1;
[X,Y]=meshgrid(x,y);
qmatrix=X.*exp(-X.^2-Y.^2);
[dx dy]=egradient(qmatrix,.2,.2);
edsymbol('spire','spire.psd');
equiver(X,Y,dx,dy,[1 1 1],'spire');
```

```
% ne
x=0.1:.2:2.1;
y=0.1:.2:2.1;
```



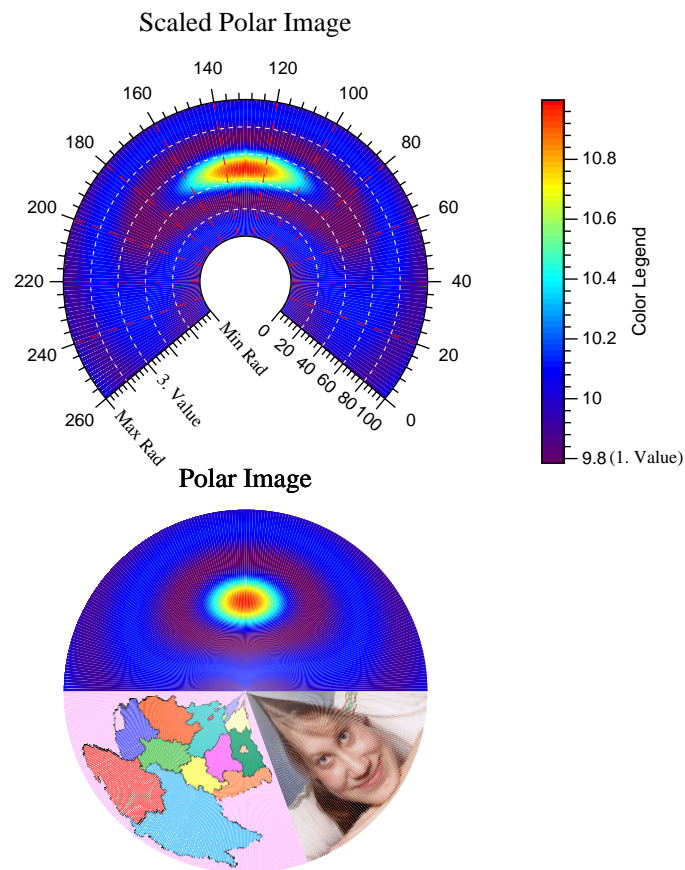
```
[X,Y]=meshgrid(x,y);
qmatrix=X.*exp(-X.^2-Y.^2);
[dx dy]=egradient(qmatrix,.2,.2);
edsymbol('needle','needle.psd');
equiver(X,Y,dx,dy,[0 0 0],'needle');

% se
x=0.1:.2:2.1;
y=-2.1:.2:-0.1;
[X,Y]=meshgrid(x,y);
qmatrix=X.*exp(-X.^2-Y.^2);
[dx dy]=egradient(qmatrix,.2,.2);
edsymbol('ftria','ftria.psd',1,0.4);
equiver(X,Y,dx,dy,[0 0 0],'ftria');

eclose(1,0);

% quiver and contour
eopen('demo5.eps')
esubepts(2,1,1,1,'demo5a.eps');
esubepts(2,1,2,1,'demo5b.eps');
eclose
if ~exist('noDemoShow')
    eview
end
% start ghostview with eps-file
```

A.1.6 edemo6.m



```

eopen('demo6.eps');
eglobpar;

% scaled polar image plot
x=-3*pi:0.1:3*pi;
[a b]=meshgrid(x,x);
R=sqrt(a.^2+b.^2) + eps;
matrix=sin(R)./R+10;
ePlotTitleText='Scaled Polar Image';
ePlotTitleDistance=15;
ePolarPlotAreaCenterPos=[70,170];
ePolarPlotAreaRadMax=40;
ePolarPlotAreaAngStart=-40;
ePolarPlotAreaAngEnd=220;
ePolarAxisRadVisible=3;
ePolarAxisRadValueVisible=1;
ePolarRadiusGridColor=[1 1 1];

```

[illegible]

A.1.7 edemo7.m

Text Features

Start at (20,210)...10mm...4mm...new font...or Symbols:αβγ

new line...go down...

go up...

and back...

redtext...
green...
blue...
red

rotate Text
rotate Text
rotate Text
rotate Text
rotate Text
rotate Text

rotate Text
rotate Text
rotate Text
rotate Text
rotate Text
rotate Text

rotate Text
rotate Text
rotate Text
rotate Text
rotate Text
rotate Text

left line1....

....center line1....

....right line1

left line2..

..center line2..

..right line2

left line3.....

.....center line3.....

.....right line3

Special Character of font 1-12 call by octal value

\41=! \42=" \43=# \44=\$ \45=% \46=& \57=/ \100=@

\133=[\134=\ \135=] \173={ \174=| \175=} \176=~

\374=ü \334=Û \344=ä \304=Ä \366=ö \326=Ö \337=ß

Special Character of font 13 call by octal value

\101=A \102=B \103=X \104=Δ \105=E \106=Φ \107=Γ ... \132=Z

\141=α \142=β \143=χ \144=δ \145=ε \146=φ \147=γ ... \172=ζ

\300=ŀ \301=Š \302=Ŕ \303=Ŗ \304= \305=⊕ \306=Ø

\307=∩ \310=∪ \311=⊃ \312=⊇ \313=⊘ \314=⊂ \315=⊆

\315=⊆ \316=€ \317=€ \320=∠ \321=∇ \322=® \323=©

```
eopen('demo7.eps');
eglobpar;
```

```
%a few global parameter
eTextFont=1;
```

```

eTextFontSize=6;

%title
etext('Text',15,230,[15*2 15 -45]);
etext('Features',0,0,[15*2 15 45]);

%append text
etext('demo',30,30,100,1,2,45,[0.9 0.9 0.9]);
etext('Start at (20,210)...',20,210);
etext('10mm...',0,0,10);
etext('4mm...',0,0,4);
etext('new font...or Symbols:',0,0,6,1,2);
etext('\\\\141\\\\142\\\\147',0,0,6,1,13);
etext('new line...',20,185);
etext('go down... ',0,-3);
etext('go up... ',0,6);
etext('and back... ',0,-3);
etext('redtext... ',0,0,6,1,5,45,[1 0 0]);
etext('yellow... ',0,0,6,1,5,-45,[1 1 0]);
etext('blue... ',0,0,6,1,5,-135,[0 0 1]);
etext('red.. ',0,0,6,1,5,-225,[1 0 0]);
etext('green. ',0,0,6,1,5,45,[0 1 0]);
colorMap=ecolors(3,5);

% rotation
etext('rotate Text',50,140,6,1,2,0,colorMap(1,:));
etext('rotate Text',50,140,6,1,2,45,colorMap(2,:));
etext('rotate Text',50,140,6,1,2,90,colorMap(3,:));
etext('rotate Text',50,140,6,1,2,135,colorMap(4,:));
etext('rotate Text',50,140,6,1,2,180,colorMap(5,:));

etext('rotate Text',100,150,6,0,2,0,colorMap(1,:));
etext('rotate Text',100,150,6,3,2,45,colorMap(2,:));
etext('rotate Text',100,150,6,3,2,90,colorMap(3,:));
etext('rotate Text',100,150,6,3,2,135,colorMap(4,:));
etext('rotate Text',100,150,6,0,2,180,colorMap(5,:));

etext('rotate Text',150,160,6,-1,2,0,colorMap(1,:));
etext('rotate Text',150,160,6,-1,2,45,colorMap(2,:));
etext('rotate Text',150,160,6,-1,2,90,colorMap(3,:));
etext('rotate Text',150,160,6,-1,2,135,colorMap(4,:));
etext('rotate Text',150,160,6,-1,2,180,colorMap(5,:));

% left center right
lineStep=-8;
yValue=130;
etext('left line1....',10,yValue,6,1);
etext('....center line1....',90,yValue,6,0);
etext('....right line1',170,yValue,6,-1);
yValue=yValue+lineStep;

```

```

etext('left line2..',10,yValue,6,1);
etext('...center line2..',90,yValue,6,0);
etext('...right line2',170,yValue,6,-1);
yValue=yValue+lineStep;
etext('left line3.....',10,yValue,6,1);
etext('.....center line3.....',90,yValue,6,0);
etext('.....right line3',170,yValue,6,-1);

%special character
eTextFont=1;
eTextFontSize=5;
xValue=10;
yValue=yValue+1.5*lineStep;
etext('Special Character of font 1-12 call by octal value',xValue,yValue);
s=' ';
for i=[33,34,35,36,37,38,47,64]
    c=sprintf('\\134%o=\\%o ',i,i);
    s=[s,c];
end
yValue=yValue+1.3*lineStep;
etext(s,xValue,yValue);
s=' ';
for i=[91,92,93,123,124,125,126]
    c=sprintf('\\134%o=\\%o ',i,i);
    s=[s,c];
end
yValue=yValue+lineStep;
etext(s,xValue,yValue);
s=' ';
for i=[252,220,228,196,246,214,223]
    c=sprintf('\\134%o=\\%o ',i,i);
    s=[s,c];
end
yValue=yValue+lineStep;
etext(s,xValue,yValue);

yValue=yValue+1.5*lineStep;
etext('Special Character of font 13 call by octal value',xValue,yValue);
yValue=yValue+1.3*lineStep;
etext(' ',xValue,yValue);
for i=65:71
    c=sprintf('\\134%o=',i);
    s=sprintf('\\%o',i);
    etext(c,0,0,eTextFontSize,1);
    etext(s,0,0,eTextFontSize,1,13);
end
etext(' ...',0,0,eTextFontSize,1);
i=90;
c=sprintf('\\134%o=',i);
s=sprintf('\\%o',i);

```

```

















etext(c,0,0,eTextFontSize,1);
etext(s,0,0,eTextFontSize,1,13);
yValue=yValue+lineStep;
etext(' ',xValue,yValue);
for i=97:103
    c=sprintf(' \\134%o=',i);
    s=sprintf(' \\%o',i);
    etext(c,0,0,eTextFontSize,1);
    etext(s,0,0,eTextFontSize,1,13);
end
etext(' ...',0,0,eTextFontSize,1);
i=122;
c=sprintf(' \\134%o=',i);
s=sprintf(' \\%o',i);
etext(c,0,0,eTextFontSize,1);
etext(s,0,0,eTextFontSize,1,13);
yValue=yValue+lineStep;
etext(' ',xValue,yValue);
for i=192:198
    c=sprintf(' \\134%o=',i);
    s=sprintf(' \\%o',i);
    etext(c,0,0,eTextFontSize,1);
    etext(s,0,0,eTextFontSize,1,13);
end
yValue=yValue+lineStep;
etext(' ',xValue,yValue);
for i=199:205
    c=sprintf(' \\134%o=',i);
    s=sprintf(' \\%o',i);
    etext(c,0,0,eTextFontSize,1);
    etext(s,0,0,eTextFontSize,1,13);
end
yValue=yValue+lineStep;
etext(' ',xValue,yValue);
for i=205:211
    c=sprintf(' \\134%o=',i);
    s=sprintf(' \\%o',i);
    etext(c,0,0,eTextFontSize,1);
    etext(s,0,0,eTextFontSize,1,13);
end

eclose;
newbbox=ebbox(5);
if ~exist('noDemoShow')
    eview % start ghostview with eps-file
end

```

Symbols



No	Filename	Symbol
1.	oplus.psd	
2.	plus.psd	
3.	star.psd	
4.	ring.psd	
5.	fring.psd	
6.	rect.psd	
7.	frect.psd	
8.	triaC.psd	
9.	ftriaC.psd	
10.	tria.psd	
11.	ftria.psd	
12.	spire.psd	
13.	farrow.psd	
14.	needle.psd	
15.	euro.psd	
16.	feuro.psd	

```
% standard plot
eopen('demo8.eps') % open eps-file and write eps-head

eglobpar % get access to global parameters
%eWinGridVisible=1;
etext('Symbols ',eWinWidth/2,eWinHeight-10,10,0)
sFiles=['oplus.psd ','plus.psd ','star.psd ','ring.psd ';...
        'fring.psd ','rect.psd ','frect.psd ','triaC.psd ';...
```



```

'ftriaC.psd';'tria.psd  ','ftria.psd ','spire.psd ';\dots
'farrow.psd';'needle.psd';'euro.psd  ','feuro.psd '];
sColors=[0.7 0.6 0.5;1 0 0;0 1 0;0 0 1;0.7 0.5 0;1 0 1;\dots
         0 0.5 1;0.5 0 1;0.6 0.3 0.2;0.8 0.5 0.7;\dots
         1 0.8 0.3;0.6 0.3 0.4;0.6 0.4 0.8;0.8 0.3 0.3;\dots
         0.6 0.6 0.3;1 0.7 0];

%define symbols
nSym=16;
for i=1:nSym
    edsymbol(sprintf('s%d',i),sFiles(i,:),... % define symbol
              1,1,0,0,0,sColors(i,:));
end

%draw symbols
nPos=40;
randPos=rand(nPos,2);
xPos(:,1)=randPos(:,1)*eWinWidth*0.7+eWinWidth*0.1;
yPos(:,2)=randPos(:,2)*eWinHeight*0.35+eWinHeight*0.56;
for i=1:nPos
    symbol=sprintf('s%d',rem(i,nSym)+1);
    esymbol(xPos(i,1),yPos(i,2),symbol,randPos(i,1)+0.4,...
            randPos(i,2)+0.3,(randPos(i,1)-randPos(i,2))*360);% draw symbol
end

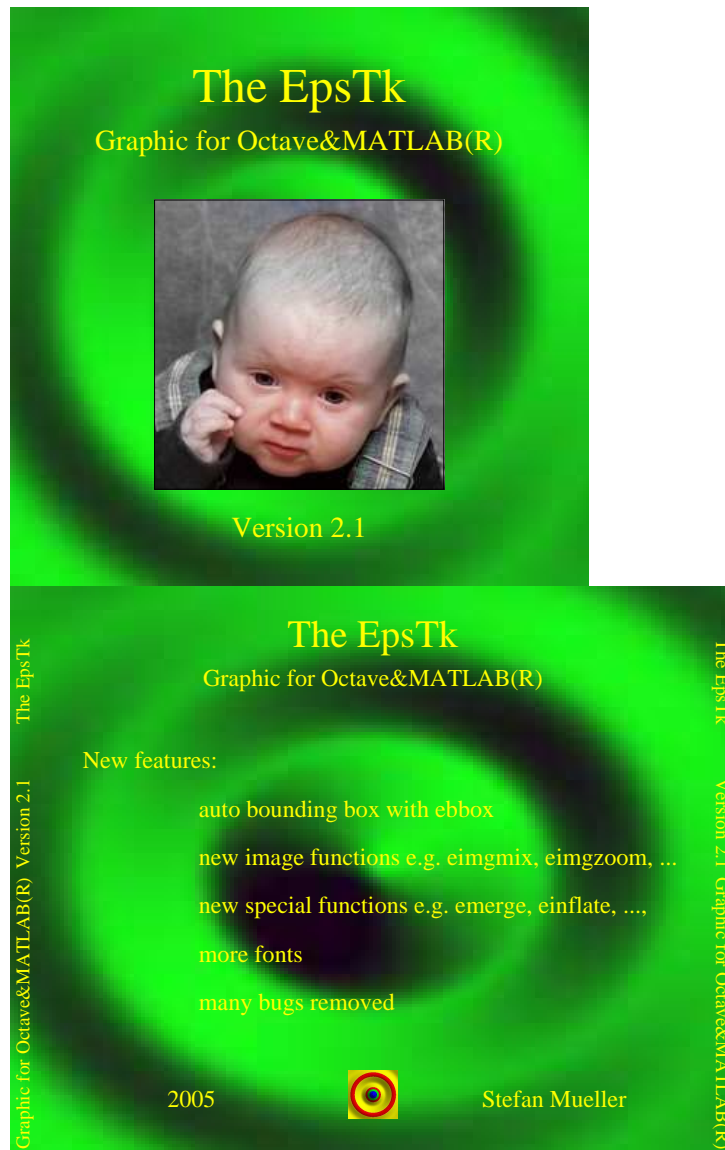
etext('Table of Symbols ',eWinWidth/2,115,10,0);

%body of table
[tabx,taby]=etabdef(nSym,3,40,0,100,100,[1 3 2]);
for i=1:nSym
    etabtext(tabx,taby,i,1,sprintf('%d.',i),-1);
    etabtext(tabx,taby,i,2,sFiles(i,:),1,1,80,[0 0 0],sColors(i,:));
    esymbol(tabx(3,1)+tabx(3,2)/2,...
            taby(i,1)+taby(i,2)/2,...
            sprintf('s%d',i),0.5,0.5);
end
etabgrid(tabx,taby);

%head of table
[htabx htaby]=etabdef(1,3,40,100,100,8,[1 3 2],1);
etabtext(htabx,htaby,1,1,'No',0,3);
etabtext(htabx,htaby,1,2,'Filename',0,3);
etabtext(htabx,htaby,1,3,'Symbol',0,3);
eclose % close eps-file
if ~exist('noDemoShow')
    eview % start ghostview with eps-file
end

```

A.1.9 edemo9.m



```
eopen('demo9.eps')           % open eps-file and write eps-head
eglobpar;

titleFile='demo_title.ppm';
backgrFile='demo_backgr.ppm';
logoFile='demo_logo.ppm';
contentFile='demo_content.txt';
```

```

%make title image
[titleImg titleCM]=eimgread([ePath 'default.jpg']); % read image
[imgH imgW]=size(titleImg);
imgMask=eimgmask(imgH,imgW,1);
imgMask=circshift(imgMask,50);
swCM=titleCM(:,1)+titleCM(:,2)+titleCM(:,3);
swCM=swCM/max(swCM);
swCM=[swCM swCM swCM]; % color -> gray
swImg=eidx2rgb(titleImg,swCM);
colorImg=eidx2rgb(titleImg,titleCM);
titleImg=eimgmix(swImg,colorImg,imgMask);
eimgwrit(titleFile,titleImg,-1); % save image

%make background image
[backImg backCM]=eshadoi; % get default shadow image
backCM(:,[1 3])=0.1; % green colormap
eppmwrit(backgrFile,backImg,backCM); % save image

%make logo image
[logoImg logoCM]=eshadois; % get default shadow image
eppmwrit(logoFile,logoImg,logoCM); % save image

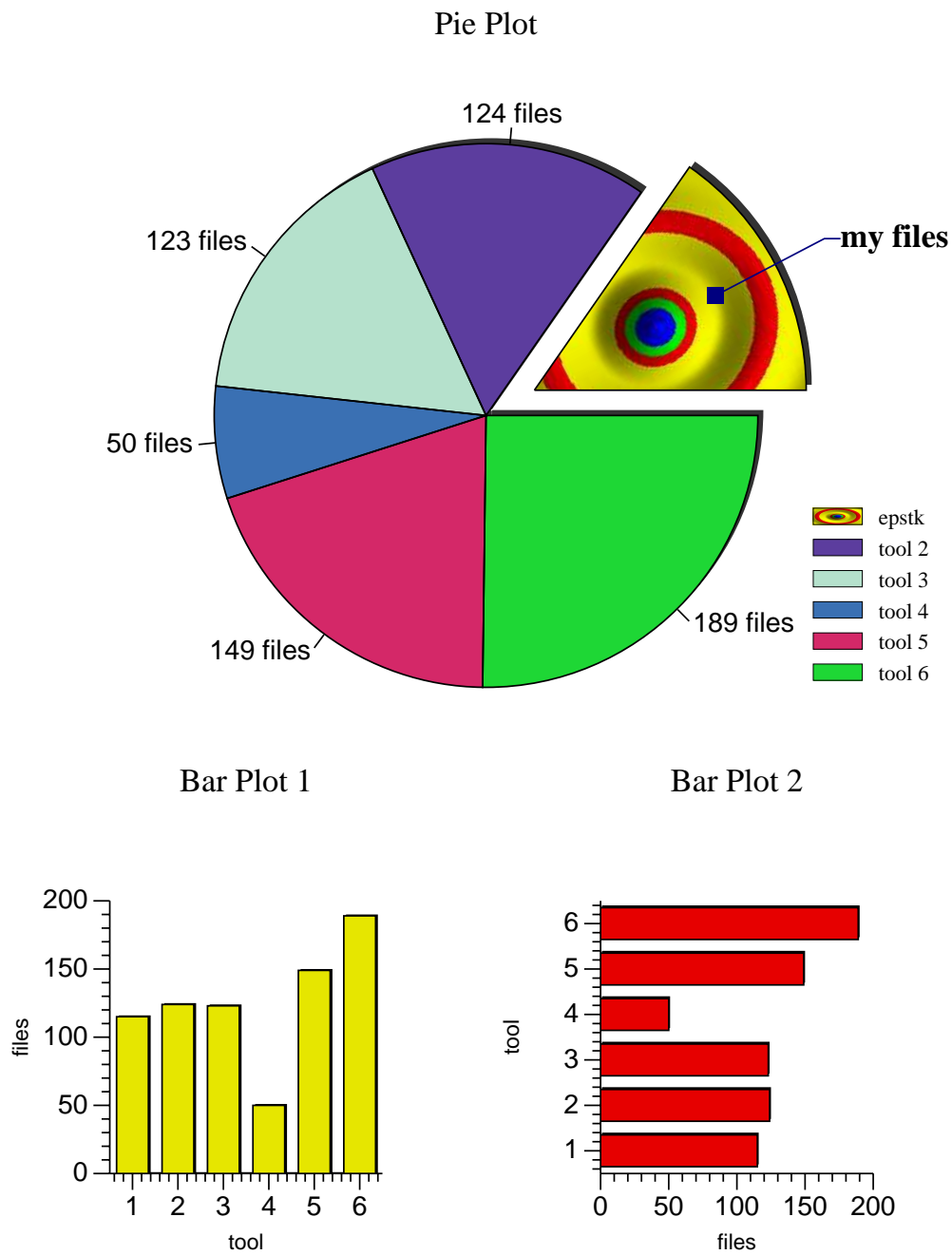
%content
lf=setstr(10); %linefeed
contenttext=[
    'New features:' lf ...
    '#auto bounding box with ebbox###' lf ...
    '#new image functions e.g. eimgmix, eimgzoom, ...###' lf ...
    '#new special functions e.g. emerge, einflate, ..., ###' lf ...
    '#more fonts###' lf ...
    '#many bugs removed###' lf ...
];
etxtwrit(contenttext,contentFile);

% make cover
ecdcover('The EpsTk',...
    'Graphic for Octave&MATLAB(R)',...
    'Stefan Mueller',...
    'Version 2.1',...
    '2005',...
    [1 1 0],...
    titleFile,backgrFile,...
    logoFile,contentFile);

eclose
if ~exist('noDemoShow')
    eview % start ghostview with eps-file
end

```

A.1.10 edemo10.m



```
eopen('demo10.eps');
```

```

eglobpar
[im icm]=eshadois;
data=[115 124 123 50 149 189];
n=length(data);
cm=ecolors(3,n);

ePolarPlotAreaRadMin=0;
ePolarPlotAreaAngEnd=360;
%data=data/sum(data)
shadowColor=[0.2 0.2 0.2];
ePlotLegendPos=[ePolarPlotAreaRadMax*2.2 30];
legend=['epstk ','tool 2','tool 3','tool 4','tool 5','tool 6'];

offset=10;

%shadow pie
epie(data(1),'',',',-1,offset,shadowColor);
for i=2:n
    epie(data(i),'',',',-1,0,shadowColor);
end
epie;

%color pie
ePolarPlotAreaCenterPos=ePolarPlotAreaCenterPos-[1 1];
eAxesValueFontSize=5;
epie(data(1),'',legend(1,:),im,offset,icm);
for i=2:n
    epie(data(i),sprintf('%d files',data(i)),legend(i,:),-1);
end
epie;

%frame pie
ePlotTitleText='Pie Plot';
epie(data(1),'',',',0,offset);
for i=2:n
    epie(data(i));
end
angles=epie;

%label
labelColor=[0 0 0.5];
labelAngle=(angles(1,1)+angles(1,2)/2)*pi/180;
p1=ePolarPlotAreaCenterPos+[cos(labelAngle) sin(labelAngle)]*...
    (offset+ePolarPlotAreaRadMax*3/4);
p2=ePolarPlotAreaCenterPos+[cos(labelAngle) sin(labelAngle)]*...
    (offset+ePolarPlotAreaRadMax+10);
p3=[ePolarPlotAreaCenterPos(1)+ePolarPlotAreaRadMax+offset+5 p2(2)];
pline=[p1;p2;p3];
epline(pline(:,1),pline(:,2),eLineWidth,0,labelColor);
edsymbol('dot','frect.psd',0.3,0.3,0,0,0,labelColor);

```

```

esymbol(p1(1),p1(2),'dot');
etext('my files',p3(1),p3(2),6,4,3);

ePlotAreaHeight=50;
ePlotAreaWidth=50;
eXAxisNorthVisible=0;
eYAxisEastVisible=0;
[xb yb]=ebar(data,0);

ePlotTitleText='Bar Plot 1';
ePlotAreaPos=[20 20];
eYAxisWestScale=[0 0 200];
eXAxisSouthScale=[0.5 0 6.5];
eYAxisWestLabelText='files';
eXAxisSouthLabelText='tool';

eplot(xb+0.05,yb+1,'',-1,[0 0 0])
eplot(xb,yb,'',-1,[0.9 0.9 0])
eplot(xb,yb,'',0,[0 0 0])
eplot

ePlotTitleText='Bar Plot 2';
ePlotAreaPos=[110 20];
eXAxisSouthScale=[0 0 200];
eYAxisWestScale=[0.5 0 6.5];
eXAxisSouthLabelText='files';
eYAxisWestLabelText='tool';
[xb yb]=ebar(data,0);
eplot(yb+1,xb+0.05,'',-1,[0 0 0])
eplot(yb,xb,'',-1,[0.9 0.0 0])
eplot(yb,xb,'',0,[0 0 0])
eplot

eclose
newbbox=ebbox(5);
if ~exist('noDemoShow')
    eview % start ghostview with eps-file
end

```

A.1.11 edemo11.m

version 2.1
epsTk News
April 2005

Econometrics with Octave

Dirk Eddebüttel* Bank of Montreal, Toronto, Canada. Dirk.Eddebuettel@bmo.com
November 1999

Summary
GNU Octave is an open-source implementation of a (mostly Matlab compatible) high-level language for numerical computations. This review briefly introduces Octave, discusses applications of Octave in an econometric context, and illustrates how to extend Octave with user-supplied C++ code. Several examples are provided.

Introduction
Econometricians sweat linear algebra. Be it for linear or non-linear problems of estimation or inference, matrix algebra is a natural way of expressing these problems on paper. However, when it comes to writing computer programs to either implement tried and tested econometric procedures, or to research and prototype new routines, programming languages such as C or Fortran are more of a burden than an aid. Having to enhance the language by supplying functions for even the most primitive operations such as transposing a matrix adds extra programming effort, introduces new points of failure, and moves the level of abstraction further away from the elegant mathematical expressions. As Eddebüttel (1996) argues, object-oriented programming provides 'a step up' from Fortran or C by enabling the programmer to seamlessly add new data types such as matrices, along with operations on these new data types, to the language. But with Moore's Law still being validated by ever and ever faster processors, and, hence, ever increasing computational power, the prime reason for using compiled code, i.e. speed, becomes less relevant. Hence the growing popularity of interpreted programming languages, both, in general, as witnessed by the surge in popularity of the generalpurpose programming languages Perl and Python and, in particular, for numerical applications with strong emphasis on matrix calculus where languages such as Gauss, Matlab, Ox, R, Splus, which were reviewed by Cribari-Neto and Jensen (1997), Cribari-Neto (1997) and Cribari-Neto and Zarkos (1999), have become popular. This article introduces another interpreted language focussed on numerical applications: Octave.

Octave is a program, as well as a programming language, for doing numerical work in a convenient yet expressive and powerful fashion. Octave was written primarily by John W. Eaton of the University of Wisconsin-Madison with various smaller contributions from other programmers across the Internet. The history of Octave can be traced back (as far as 1988) to the idea of providing companion software for a textbook in chemical engineering, written by two researchers from the University of Wisconsin-Madison and the University of Texas. Full-time development for Octave began in the spring of 1992.

Version 1.0 was released in February of 1994, and version 2.0 followed in January of 1997. Octave is part of the GNU project (which is on the Web at <http://www.gnu.org>) and released under the GNU General Public License (GPL). It can therefore be used, studied, extended or simply copied just like any other Free Software program.

Econometricians might want to employ Octave for a variety of applications as it provides a convenient tool for interactive work which can range from simple applications to ad hoc simulations. The user interface, i.e. the Octave shell, is flexible and powerful. The rich set of mathematical operators and functions make Octave a very good prototyping and programming platform, or data pre- and postprocessing application. As Octave can be extended with new C, C++ or Fortran code that runs at the speed of compiled code, it is a good building block for Monte Carlo simulations. Octave is very similar to Matlab, and to a lesser extent to Gauss. However, both of these are commercial 'closed source' programs whereas Octave, as noted above, is distributed under the GNU GPL.

... Octave also contains plotting commands which are implemented as wrappers around the external Gnuplot program. These commands include a generic plot command for plotting one or several series, as well as commands for plotting histograms and bar charts. All of these can be annotated, and it is possible to have several plot windows open at the same time (which requires a Gnuplot binary more recent than the 3.5 releases). The Octave plot commands are very convenient, but clearly not as complete as the Matlab plotting functions and hence might not satisfy all needs for publication-quality graphs. However, add-on packages have been provided by Octave users. Debian GNU/Linux provides additional graphics packages such as octave-plplot, an Octave interface to the PLPlot program, and

M-Files of epsTk

eaxes.m	egridpol.m	equiver.m
eaxespol.m	egridpr.m	erect.m
eaxis.m	egridxy.m	erectrc.m
ebars.m	ehad.m	erencode.m
ebbox.m	eidx2rgb.m	erespar.m
ebitmap.m	eimage.m	erespar2.m
ebright.m	eimagesc.m	ergb2idx.m
ecdcovr.m	eimagexy.m	erbsplitt.m
echrcode.m	eimg2txt.m	esavpar.m
eclip.m	eimgleg.m	esavpar2.m
eclippol.m	eimgmask.m	escalecl.m
eclose.m	eimgmix.m	escalelog.m
ecolors.m	eimgread.m	escalepa.m
econtour.m	eimgrot.m	escalexy.m
econtra.m	eimgview.m	eshadoi.m
edemo1.m	eimgwrit.m	eshadoix.m
edemo10.m	eimgzoom.m	eshadow.m
edemo11.m	einflate.m	esubeps.m
edemo12.m	einit.m	esymbol.m
edemo13.m	einseps.m	etabdef.m
edemo14.m	eisoline.m	etabgrid.m
edemo15.m	ejpg2eps.m	etabtext.m
edemo16.m	ejpglist.m	etext.m
edemo2.m	ejpgread.m	etextxy.m
edemo3.m	elinelip.m	etextdis.m
edemo4.m	elines.m	etitle.m
edemo5.m	emerge.m	etxt2box.m
edemo6.m	eopen.m	etxtbox.m
edemo7.m	eparam.m	etxtlpos.m
edemo8.m	epic.m	etxtread.m
edemo9.m	epiesxy.m	etxtwrit.m
edemos.m	epline.m	eusage.m
edsymbol.m	eplo2win.m	evect2a85.m
eellipse.m	eplot.m	eview.m
eellipxy.m	eplotlg.m	ewinsize.m
eerrbar.m	epolar.m	exyline.m
efillmat.m	epolari.m	exyplot.m
eframe.m	epolaris.m	exyplotc.m
eglobpar.m	epolpof.m	exyplotf.m
egradient.m	epolplos.m	exyploti.m
egrid.m	epolplot.m	exyplots.m
egridcl.m	eppmread.m	
egridlog.m	eppmwrit.m	
egridpa.m	eptitle.m	



New feature of 2.1

- auto bounding box with
- new image functions e.g. eimgmix, eimgzoom, ...
- new special functions e.g. emerge, einflate, ...
- more fonts
- many of bugs removed

octave-epstk, a package for the direct creation of encapsulated postscript graphs.

...

```
bgcolor=[1 1 0.8];
eopen('demo11.eps');
eglobpar
```

```

% head
%eWinGridVisible=1;
w=eWinWidth;
fs=20;
x=0;
h=fs/2;y=eWinHeight-h;
text='epsTk News';
eframe(x,y,w,h,0,-1,[0.0 0.5 0.0]); %background
etxtbox(text,x,y,w,h,[fs fs 10],0,1,0,[0 0 0],[1 1]); %shadow text
etxtbox(text,x,y,w,h,[fs],0,1,0,bgcolor); %main text
etext('April 2005',160,y+h/2,4,3,1,0,[1 1 0]); %text
h=h*0.8;y=y-h;
eframe(x,y,w,h,0,-1,bgcolor); %background
etxtbox(text,x,y,w,h,[fs fs 10],0,1,0,[0 0 0],[1 1+fs/2]); %shadow text
etxtbox(text,x,y,w,h,fs,0,1,0,[1 0 0],[0 fs/2]); % main text
yHead=y;
eframe(0,0,eWinWidth,yHead,0,-1,bgcolor); %background of ellipse
eellipse(x+20,y+h,40,10,0,-1,[1 0 0],15); %fill ellipse
eellipse(x+20,y+h,40,10,0,0,[0 0 0],15); %border of ellipse
etext('version 2.1',x+20,y+h,4,3,14,15,[1 1 0]); %text

% text
eTextBoxSpaceWest=1.5;
eTextBoxSpaceEast=1.5;
eTextBoxSpaceNorth=1.5;
w=eWinWidth/3;
h=17;y=y-h;
fs=6;
text='Econometrics with Octave';
etxtbox(text,x,y,w,h,[fs*1.5 fs 10],0,3,0,[0 0 0]); %title

rText=etxtread([ePath 'octave.asc']); %get text from file
h=y;y=y-h;
fs=3;
endKey='Version 1.0';
pos=findstr(rText,endKey);
text=rText(1:pos(1)-1);
rText=rText(pos(1):length(rText));
etxtbox(text,x,y,w,h,fs,2,1,0,[0 0 0]); %1. column of text

x=x+w;
h=143;y=yHead-h;
endKey='octave-epstk';
pos=findstr(rText,endKey);
text=rText(1:pos(1)-1);
rText=rText(pos(1):length(rText));
etxtbox(text,x,y,w,h,fs,2,1,0,[0 0 0]); %2. column of text

[im cm]=eshadois;

```



```

h=w;y=y-h;
eframe(x,y,w,w,0.5,im,cm); % insert epstk logo
eframe(x,y,w,w,0.5,0,[0 0 0]); %frame

h=50;y=y-h;
text=rText;
etxtbox(text,x,y,w,h,fs,2,3,0,[1 0 0]); % last part of text

% filelist
fs=6;
x=x+w;
h=eTextBoxSpaceNorth+fs+eTextBoxSpaceSouth;y=yHead-h;
text='M-Files of epsTk';
etxtbox(text,x,y,w,h,fs,0,3,0,[0 0 0]); %title
text=etxtread([ePath 'mFileList']);
[tPos n]=etxtlpos(text);
nRows=ceil(n/3);
fs=3.2;
h=eTextBoxSpaceNorth+nRows*fs+eTextBoxSpaceSouth;y=y-h;
cx=x;
cText=text(tPos(1,1):tPos(nRows,2));
etxtbox(cText,cx,y,w/3,h,fs,2,1,0,[0 0 1]); %1. column
cx=x+w/3;
cText=text(tPos(nRows+1,1):tPos(2*nRows,2));
etxtbox(cText,cx,y,w/3,h,fs,2,1,0,[0 0 1]); %2. column
cx=x+2*w/3;
cText=text(tPos(2*nRows+1,1):tPos(n,2));
etxtbox(cText,cx,y,w/3,h,fs,2,1,0,[0 0 1]); %3. column

% feature list
y=y-fs;
fs=6;
h=eTextBoxSpaceNorth+fs+eTextBoxSpaceSouth;y=y-h;
text='New feature of 2.1';
etxtbox(text,x,y,w,h,fs,0,3,0,[0 0 0]); %title

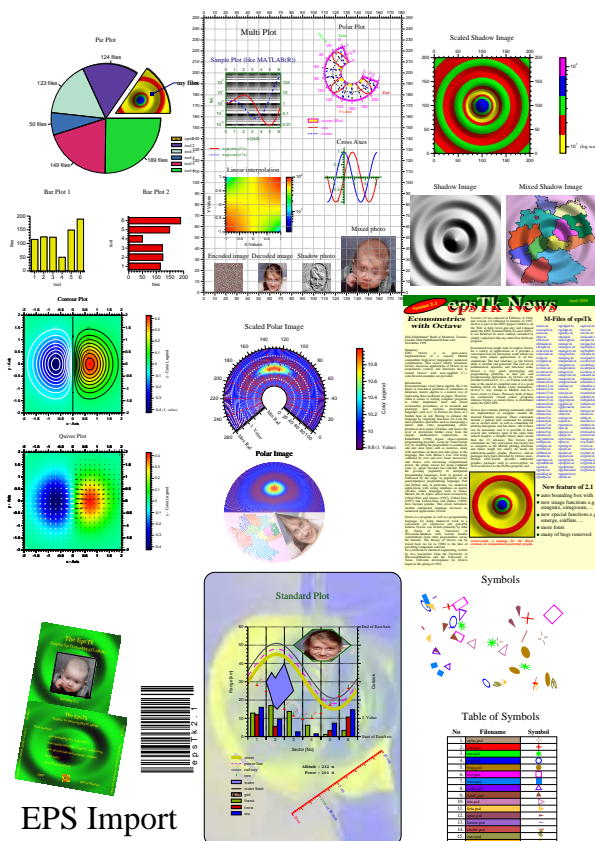
fs=5;
font=1;
dotSize=fs/2;
iDotX=x+eTextBoxSpaceWest+dotSize/2;
iDotShift=eTextBoxSpaceNorth+fs/2;
iTxtX=iDotX+dotSize/2;
iw=w-iTxtX+x;
edsymbol('dot','fring.psd',dotSize/10,dotSize/10,0,0,0,[0 0.3 0]);

rows=1;h=eTextBoxSpaceNorth+rows*fs+eTextBoxSpaceSouth;y=y-h;
esymbol(iDotX,y+h-iDotShift,'dot');
text='auto bounding box with ebbbox';
etxtbox(text,iTxtX,y,iw,h,fs,1,font,0,[0 0 0]); %feature list item

```

[illegible]

A.1.12 `edemo12.m`



```

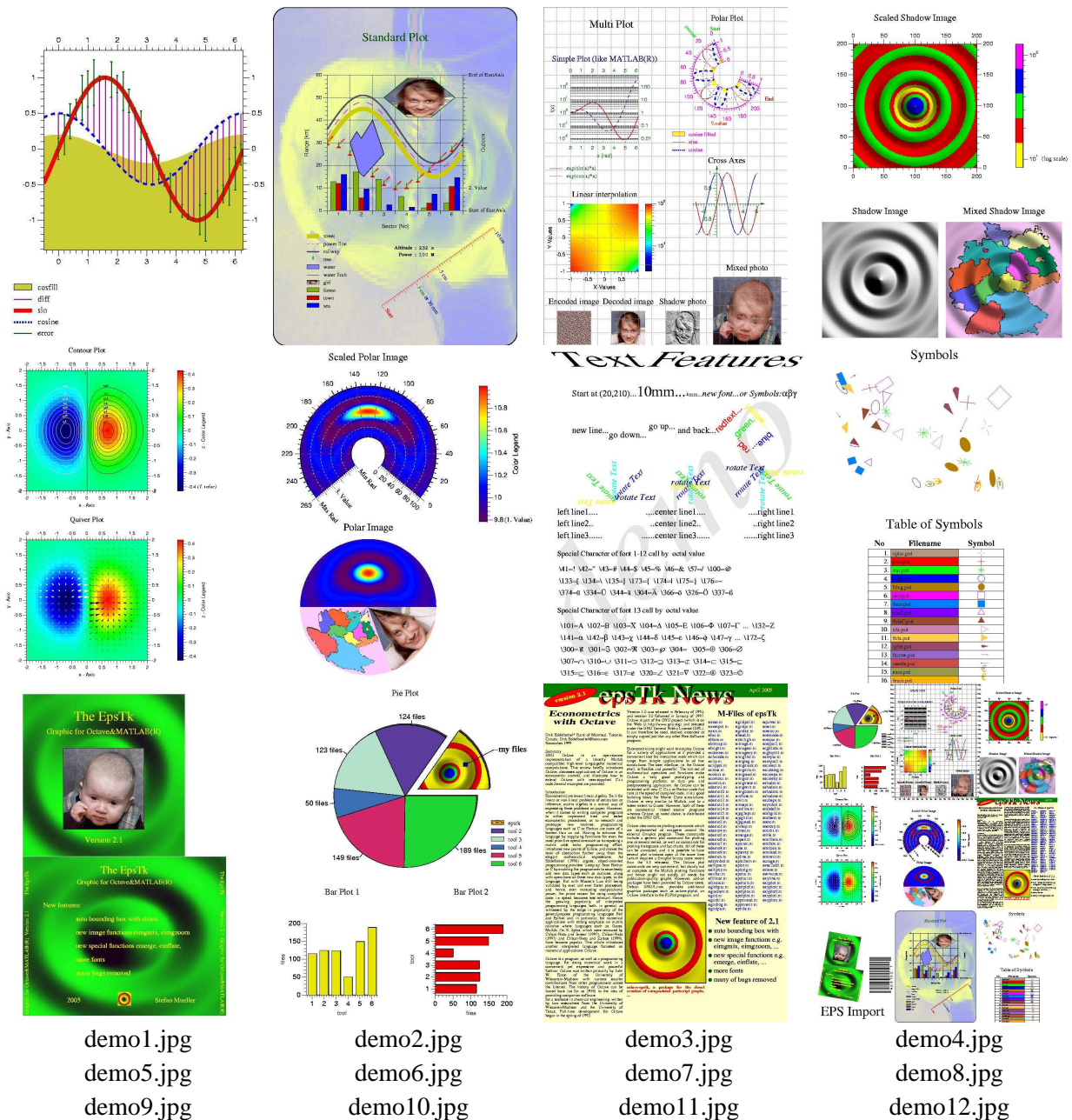
eopen('demo12.eps') % open eps-file and write eps-head

eglobpar % get access to global parameters
etext('EPS Import',5,5,10,1)

esubeps(3,3,1,1,'demo10.eps');
esubeps(3,3,1,2,'demo3.eps');
esubeps(3,3,1,3,'demo4.eps');
esubeps(3,3,2,1,'demo5.eps');
esubeps(3,3,2,2,'demo6.eps');
esubeps(3,3,2,3,'demo11.eps');
einseps(0,20,'demo9.eps',0.2,0.2,-10);
einseps(60,20,[ePath 'epstkbk'],0.6,0.6,90); % file generated by GNU barcode
esubeps(3,3,3,2,'demo2.eps');
esubeps(3,3,3,3,'demo8.eps');
eclose % close eps-file
if ~exist('noDemoShow')
    eview % start ghostview with eps-file
end

```

A.1.13 edemo13.m



```
% create a image list with ejpglist()
```

```
eglobpar
einit

% read  eps file list
list=etxtread([ePath 'epsFileList']);
jList=' ';
[lPos n]=etxtlpos(list);

% make jpeg-files and a list of jpeg-files
for i=1:n
    eFileName=list(lPos(i,1):lPos(i,2));
    if exist(eFileName)
        jFileName=ebitmap(1,100); % eps-file to jpeg-file
        jList=[jList jFileName eTextLimitPara];
    end
end
jList=jList(2:size(jList,2));
etxtwrit(jList,'demo_jFileList');

% create image list by ejpglist()
ejpglist('demo_jFileList',[50 40],0,'demo13');
ebox(5,'demo13.eps');
if ~exist('noDemoShow')
    eview % start ghostview with eps-file
end
```

A.1.14 edemo14.m

Januar	Februar	März	April	Mai	Juni
1 Sa Neujahr	1 Di	1 Di	1 Fr	1 So 1. Mai	1 Mi
2 So	2 Mi	2 Mi	2 Sa	2 Mo	2 Do
3 Mo	3 Do	3 Do	3 So	3 Di	3 Fr
4 Di	4 Fr	4 Fr	4 Mo	4 Mi	4 Sa
5 Mi	5 Sa	5 Sa	5 Di	5 Do Chr. Himmelf.	5 So
6 Do	6 So Fastnacht	6 So	6 Mi	6 Fr vorgearb.	6 Mo
7 Fr	7 Mo Fastnacht	7 Mo	7 Do	7 Sa	7 Di
8 Sa	8 Di	8 Di	8 Fr	8 So	8 Mi
9 So	9 Mi	9 Mi	9 Sa	9 Mo	9 Do km
10 Mo	10 Do	10 Do	10 So	10 Di	10 Fr
11 Di	11 Fr	11 Fr	11 Mo	11 Mi	11 Sa
12 Mi	12 Sa	12 Sa	12 Di	12 Do gle	12 So
13 Do	13 So	13 So	13 Mi	13 Fr	13 Mo
14 Fr	14 Mo	14 Mo	14 Do	14 Sa	14 Di
15 Sa	15 Di	15 Di	15 Fr	15 So Pfingsten	15 Mi
16 So	16 Mi	16 Mi	16 Sa	16 Mo Pfingsten	16 Do
17 Mo	17 Do	17 Do	17 Sa	17 Di	17 Fr
18 Di	18 Fr	18 Fr	18 Mo	18 Mi	18 Sa
19 Mi	19 Sa	19 Sa	19 Di	19 Do	19 So
20 Do	20 So	20 So	20 Mi	20 Fr	20 Mo
21 Fr	21 Mo	21 Mo	21 Do	21 Sa	21 Di mw
22 Sa	22 Di	22 Di	22 Fr	22 So	22 Mi
23 So	23 Mi	23 Mi	23 Sa	23 Mo	23 Do kus
24 Mo	24 Do	24 Do	24 So	24 Di	24 Fr
25 Di	25 Fr	25 Fr Karfreitag	25 Mo	25 Mi	25 Sa
26 Mi	26 Sa	26 Sa	26 Di	26 Do Fronleich.	26 So
27 Do	27 So	27 So Ostern	27 Mi	27 Fr vorgearb.	27 Mo
28 Fr	28 Mo	28 Mo Ostern	28 Do	28 Sa	28 Di
29 Sa	29 Mi	29 Mi	29 Fr	29 So	29 Mi
30 So	30 Do	30 Do	30 Sa	30 Mo	30 Do
31 Mo	31 Di	31 Di	31 Mi	31 Do	31 Do

ANNO 2005

Juli	August	September	Oktober	November	Dezember
1 Fr	1 Mo	1 Do	1 Sa	1 Di Allerheiligen sk.	1 Do
2 Sa	2 Di	2 Fr	2 So	2 Mi	2 Fr
3 So	3 Mi	3 Sa	3 Mo Dr. Einheit	3 Do	3 Sa
4 Mo	4 Do	4 So	4 Di	4 Fr	4 So
5 Di	5 Fr	5 Mo	5 Mi	5 Sa	5 Mo
6 Mi	6 Sa	6 Di	6 Do	6 So	6 Di
7 Do	7 So	7 Mi	7 Fr	7 Mo	7 Mi
8 Fr	8 Mo	8 Do	8 Sa	8 Di js	8 Do bra
9 So	9 Di	9 Fr	9 So	9 Mi	9 Fr
10 Mo	10 Mi	10 Sa	10 Mo	10 Do	10 So
11 Di	11 Do	11 So	11 Di	11 Fr	11 So ros
12 Mi	12 Fr	12 Mo	12 Mi	12 Sa	12 Mo
13 Do	13 So	13 Di	13 Do	13 So	13 Di
14 Mo	14 So	14 Mi	14 Fr	14 Mo	14 Mi
15 Di	15 Mo	15 Do	15 Sa	15 Di	15 Do
16 So	16 Di	16 Fr	16 So	16 Mi	16 Fr
17 Mo	17 Mi	17 Sa	17 Mo	17 Do	17 Sa
18 Di	18 Do	18 So	18 Di	18 Fr	18 So
19 Mi	19 Fr	19 Mo	19 Mi	19 Sa	19 Mo
20 Do	20 So	20 Di	20 Do	20 So	20 Di
21 Fr	21 Mo	21 Mi	21 Fr	21 Mo	21 Mi
22 Sa	22 Do	22 Do	22 So	22 Di kus	22 Do
23 So	23 Di	23 Fr	23 So	23 Mi	23 Fr
24 Mo	24 Mi	24 Sa	24 Mo	24 Do	24 Sa
25 Do	25 So	25 So	25 Di	25 Fr	25 So Weihnachten
26 Di	26 Fr	26 Mo	26 Mi	26 Sa	26 Mo Weihnachten
27 Mi	27 Sa	27 Di	27 Do	27 So	27 Di vorgearb.
28 Do	28 So	28 Mi	28 Fr	28 Mo	28 Mi vorgearb.
29 Fr	29 Mo	29 Do	29 Sa	29 Di	29 Do vorgearb. can
30 Sa	30 Di	30 Fr	30 So	30 Mi	30 Fr vorgearb.
31 So	31 Mi	31 Do	31 Mo vorgearb.	31 Do	31 Sa Silvester

```
% print calendar of a year
```

```
% written by Coletta Schumacher and stefan.mueller@fgan.de (C) 2005
```

```
year=2005;
```

```
%
myHolidays= [ day month textIndex textColumn textColor backgroundColor
                29 1 01 2 0.0 0.0 0.0 0.9 0.9 0.0;
                8 2 02 2 0.0 0.0 0.0 0.9 0.9 0.0;
                23 2 03 2 0.0 0.0 0.0 0.9 0.9 0.0;
                13 3 04 2 0.0 0.0 0.0 0.9 0.9 0.0;
                15 3 05 2 0.0 0.0 0.0 0.9 0.9 0.0;
```

14	4	06	2	0.0 0.0 0.0	0.9 0.9 0.0;
12	5	07	2	0.0 0.0 0.0	0.9 0.9 0.0;
9	6	08	2	0.0 0.0 0.0	0.9 0.9 0.0;
21	6	09	2	0.0 0.0 0.0	0.9 0.9 0.0;
23	6	10	2	0.0 0.0 0.0	0.9 0.9 0.0;
3	7	11	2	0.0 0.0 0.0	0.9 0.9 0.0;
12	7	12	2	0.0 0.0 0.0	0.9 0.9 0.0;
13	7	13	2	0.0 0.0 0.0	0.9 0.9 0.0;
14	7	14	2	0.0 0.0 0.0	0.9 0.9 0.0;
24	7	15	2	0.0 0.0 0.0	0.9 0.9 0.0;
25	7	16	2	0.0 0.0 0.0	0.9 0.9 0.0;
27	7	17	2	0.0 0.0 0.0	0.9 0.9 0.0;
9	8	18	2	0.0 0.0 0.0	0.9 0.9 0.0;
13	8	19	2	0.0 0.0 0.0	0.9 0.9 0.0;
14	9	20	2	0.0 0.0 0.0	0.9 0.9 0.0;
8	10	21	2	0.0 0.0 0.0	0.9 0.9 0.0;
16	10	22	2	0.0 0.0 0.0	0.9 0.9 0.0;
22	10	23	2	0.0 0.0 0.0	0.9 0.9 0.0;
26	10	24	2	0.0 0.0 0.0	0.9 0.9 0.0;
1	11	25	2	0.0 0.0 0.0	0.9 0.9 0.0;
8	12	26	2	0.0 0.0 0.0	0.9 0.9 0.0;
11	12	27	2	0.0 0.0 0.0	0.9 0.9 0.0;
29	12	28	2	0.0 0.0 0.0	0.9 0.9 0.0;
6	5	80	1	0.0 0.0 0.0	0.9 0.8 0.5;
27	5	80	1	0.0 0.0 0.0	0.9 0.8 0.5;
31	10	80	1	0.0 0.0 0.0	0.9 0.8 0.5;
27	12	80	1	0.0 0.0 0.0	0.9 0.8 0.5;
28	12	80	1	0.0 0.0 0.0	0.9 0.8 0.5;
29	12	80	1	0.0 0.0 0.0	0.9 0.8 0.5;
30	12	80	1	0.0 0.0 0.0	0.9 0.8 0.5;

];

myHoliText= [

'001pro	';
'002stm	';
'003rsc	';
'004tbr	';
'005do	';
'006lu	';
'007gle	';
'008krm	';
'009mwi	';
'010kos	';
'011jsh	';
'012mue	';
'013wor	';
'014us	';
'015wal	';
'016bal	';
'017gel	';
'018hec	';

```

'019sch      ';
'020mat      ';
'021js       ';
'022bck      ';
'023kus      ';
'024file     ';
'025stk      ';
'026bra      ';
'027ros      ';
'028caa      ';
'080vorgearb.  ';
];
%variable holydays
%      day    month  textIndex textColumn textColor    backgroundColor
varHolidays= [
    0     0     101      1          1.0 1.0 1.0    1.0 0.0 0.0;
    1     0     101      1          1.0 1.0 1.0    1.0 0.0 0.0;
   47     0     103      1          1.0 1.0 1.0    1.0 0.0 0.0;
   49     0     104      1          1.0 1.0 1.0    1.0 0.0 0.0;
   50     0     105      1          1.0 1.0 1.0    1.0 0.0 0.0;
   88     0     106      1          1.0 1.0 1.0    1.0 0.0 0.0;
   98     0     107      1          1.0 1.0 1.0    1.0 0.0 0.0;
   99     0     107      1          1.0 1.0 1.0    1.0 0.0 0.0;
  109     0     109      1          1.0 1.0 1.0    1.0 0.0 0.0;
];
varHoliText= [
    '101Fastnacht      ';
    '103Karfreitag     ';
    '104Ostern          ';
    '105Ostern          ';
    '106Chr.Himmelf.   ';
    '107Pfingsten       ';
    '109Fronleich.     ';
];
% fixed holidays
%      day    month  textIndex textColumn textColor    backgroundColor
fixHolidays=[
    1     1     210      1          1.0 1.0 1.0    1.0 0.0 0.0;
    1     5     211      1          1.0 1.0 1.0    1.0 0.0 0.0;
    3    10     212      1          1.0 1.0 1.0    1.0 0.0 0.0;
    1    11     213      1          1.0 1.0 1.0    1.0 0.0 0.0;
   25    12     214      1          1.0 1.0 1.0    1.0 0.0 0.0;
   26    12     214      1          1.0 1.0 1.0    1.0 0.0 0.0;
   31    12     216      1          1.0 1.0 1.0    1.0 0.0 0.0;
];
fixHoliText=[
    '210Neujahr        ';
    '2111.Mai          ';
    '212Dt. Einheit    ';
    '213Allerheiligen';
];

```



```

        '214Weihnachten ';
        '216Silvester   ';
    ];
weekday= ['Mo'; 'Di'; 'Mi'; 'Do'; 'Fr'; 'Sa'; 'So'];
saBgColor=[0.8 0.8 1.0];
suBgColor=[0.7 0.7 1.0];
monthT=['Januar   '; 'Februar  '; 'M\344rz '; 'April   ';
        'Mai      '; 'Juni     '; 'Juli    '; 'August  ';
        'September'; 'Oktober  '; 'November'; 'Dezember'];
nDaysOfM = [31 28 31 30 31 30 31 31 30 31];
if ~rem (year,4),nDaysOfM(2)=29;end
nDaysOfY=sum(nDaysOfM);

% sundays of carneval until 2019
cSundays=[ 5 3;25 2;10 2; 2 3;22 2; 6 2;26 2;18 2; 3 2;22 2;...
          14 2; 6 3;19 2;10 2; 2 3;15 2; 7 2;26 2;11 2; 3 3];
cSunday=year-2000+1;
cDay=cSundays(cSunday,1);
cMonth=cSundays(cSunday,2);
dayOfY=rem(cDay+sum(nDaysOfM(1:cMonth-1)),7);
if dayOfY
    firstDayOfY=7-dayOfY+1;
else
    firstDayOfY=1;
end

% variable holidays   day of the
for k=1:size(varHolidays,1)
    varDay=cDay+varHolidays(k,1);
    for i=0:4
        if varDay>nDaysOfM(cMonth+i)
            varDay=varDay-nDaysOfM(cMonth+i);
        else
            break
        end
    end
    varHolidays(k,1)=varDay;
    varHolidays(k,2)=cMonth+i;
end

holidays=[myHolidays;varHolidays;fixHolidays];
holitext=[myHoliText;varHoliText;fixHoliText];
[nTextRows nTextCols]=size(holitext);
holiIndex=holidays(:,3);
for i=1:nTextRows
    index=str2num(holitext(i,1:3));
    fresult=find(holiIndex==index);
    holidays(fresult,3)=i*ones(size(fresult,1),1);
end
holitext=holitext(:,4:nTextCols);

```

```

% draw table
eopen('demo14.eps');
eglobpar
eWinGridVisible=0;
dayOfY=0;
[calX calY]=etabdef(32,6,0,130,180,120);
for month=1:12
    if month==7
        etabgrid(calX,calY);
        [calX calY]=etabdef(32,6,0,0,180,120);
    end
    tabCol=rem(month-1,6)+1;
    etabtext(calX,calY,1,tabCol,monthT(month,:),0,3,100,[1 1 1],[0.0 0.5 0.0]);
    offset=3.8;
    [dayX dayY]=etabdef(32,1,calX(tabCol,1)+offset,calY(32,1),1,120);
    [wdX wdY]=etabdef(32,1,calX(tabCol,1)+0.9*offset,calY(32,1),1,120);
    for dayOfM=1:nDaysOfM(month)
        dayOfW=rem(firstDayOfY-1+dayOfY,7)+1;
        dayOfY=dayOfY+1;
        if dayOfW==6
            etabtext(calX,calY,dayOfM+1,tabCol,'',1,1,100,[1 1 1],saBgColor);
        elseif dayOfW==7
            etabtext(calX,calY,dayOfM+1,tabCol,'',1,1,100,[1 1 1],suBgColor);
        end
        etabtext(dayX,dayY,dayOfM+1,1,sprintf('%d',dayOfM),-1);
        etabtext(wdX,wdY,dayOfM+1,1,sprintf('%s',weekday(dayOfW,:)),1,3,70);
    end
    offset=8;
    [nX nY]=etabdef(32,2,calX(tabCol,1)+offset,calY(32,1),...
        calX(tabCol,2)-offset,120,[3 1]);
    for notes=find(holidays(:,2)==month)
        if holidays(notes,4)==1
            etabtext(nX,nY,holidays(notes,1)+1,1,...
                sprintf('%s',holitext(holidays(notes,3,:))),...
                1,1,100,holidays(notes,5:7),holidays(notes,8:10));
        elseif holidays(notes,4)==2
            etabtext(nX,nY,holidays(notes,1)+1,2,...
                sprintf('%s',holitext(holidays(notes,3,:))),...
                1,1,80,holidays(notes,5:7),holidays(notes,8:10));
        end
    end
end
etabgrid(calX,calY);
etext(sprintf('ANNO %d',year),90,122,8,0,3);
eclose;
if ~exist('noDemoShow')
    eview % start ghostview with eps-file
end

```

A.1.15 edemo15.m

Font List of epsTk

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 1
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 2
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 3
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 4
 ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 5
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 6
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 7
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 8
 ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 9
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 10
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 11
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 12
 ΑΒΧΔΕΦΓΗΘΚΑΜΝΟΠΘΡΣΤΥςΩΞΨΖαβχδεφγηιφκλμνοπθρστυωξψζ Φοντ No.: 13
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 14
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 15
 ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 16
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 17
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 18
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 19
 ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 20
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 21
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 22
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 23
 ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 24
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 25
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 26
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 27
 ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 28
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 29
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 30
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 31
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz Font No.: 32

[illegible]

93

Text Image

```
% text image
[im cm]=eimgread;
text=eimg2txt(im,cm);
```

```
[rows cols]=size(text);
eopen('demo16.eps');
eglobpar
etitle('Text Image',25,9);
headSpace=30;
lineSpace=1.2;
fontSize=1.5;
for i=1:rows
    y=eWinHeight-headSpace-i*fontSize*lineSpace;
    etext(text(i,:),0+eps,y,fontSize,1,11);
end
eclose
ebbox(5);
if ~exist('noDemoShow')
    eview % start ghostview with eps-file
end
```

A.2 Character Code

Character Codes

Character codes of font 1-12 call by octal value

```
\0=    \1=    \2=    \3=    \4=    \5=    \6=    \7=    \10=   \11=   \12=   \13=   \14=   \15=   \16=
\17=   \20=   \21=   \22=   \23=   \24=   \25=   \26=   \27=   \30=   \31=   \32=   \33=   \34=   \35=
\36=   \37=   \40=   \41=!   \42="   \43=#   \44=$   \45=%   \46=&   \47='   \50=(   \51=)   \52=*   \53=+   \54=,
\55=-   \56=.   \57=/   \60=0   \61=1   \62=2   \63=3   \64=4   \65=5   \66=6   \67=7   \70=8   \71=9   \72=:   \73=;
\74=<   \75==   \76=>   \77=?   \100=@  \101=A  \102=B  \103=C  \104=D  \105=E  \106=F  \107=G  \110=H  \111=I  \112=J
\113=K  \114=L  \115=M  \116=N  \117=O  \120=P  \121=Q  \122=R  \123=S  \124=T  \125=U  \126=V  \127=W  \130=X  \131=Y
\132=Z  \133=[  \134=\  \135=]  \136=^  \137=_  \140='   \141=a  \142=b  \143=c  \144=d  \145=e  \146=f  \147=g  \150=h
\151=i  \152=j  \153=k  \154=l  \155=m  \156=n  \157=o  \160=p  \161=q  \162=r  \163=s  \164=t  \165=u  \166=v  \167=w
\170=x  \171=y  \172=z  \173={  \174=|  \175=}  \176=~  \177=  \200=  \201=  \202=  \203=  \204=  \205=  \206=
\207=  \210=  \211=  \212=  \213=  \214=  \215=  \216=  \217=  \220=  \221=  \222=  \223=  \224=  \225=
\226=  \227=  \230=  \231=  \232=  \233=  \234=  \235=  \236=  \237=  \240=  \241=¡  \242=¢  \243=£  \244=¥
\245=¥  \246=f  \247=§  \250=ª  \251='   \252="   \253=«   \254=‹   \255=›   \256=fi  \257=fl  \260=°  \261=—  \262=†  \263=‡
\264=·  \265=µ  \266=¶  \267=•  \270=,  \271=..  \272="   \273=»  \274=...  \275=%  \276=¾  \277=¿  \300=À  \301=`  \302='
\303=^  \304=Ä  \305=^-  \306=˘  \307=˙  \310=¨  \311=É  \312=°  \313=„  \314=İ  \315=˝  \316=˚  \317=˘  \320=—  \321=Ñ
\322=Ò  \323=Ó  \324=Ô  \325=Õ  \326=Ö  \327=×  \330=Ø  \331=Ù  \332=Ú  \333=Û  \334=Ü  \335=Ý  \336=Þ  \337=ß  \340=à
\341=Æ  \342=â  \343=ª  \344=ä  \345=å  \346=æ  \347=ç  \350=Ł  \351=Ø  \352=Œ  \353=°  \354=ı  \355=ı  \356=ı  \357=ı
\360=ð  \361=æ  \362=ò  \363=ó  \364=ô  \365=ı  \366=ö  \367=÷  \370=ı  \371=ø  \372=œ  \373=ß  \374=ü  \375=  \376=p
```

Character Codes of font 13 call by octal value

```
\0=    \1=    \2=    \3=    \4=    \5=    \6=    \7=    \10=   \11=   \12=   \13=   \14=   \15=   \16=
\17=   \20=   \21=   \22=   \23=   \24=   \25=   \26=   \27=   \30=   \31=   \32=   \33=   \34=   \35=
\36=   \37=   \40=   \41=!   \42=∇  \43=#   \44=∑  \45=%   \46=&   \47=ε  \50=(   \51=)   \52=*   \53=+   \54=,
\55=-   \56=.   \57=/   \60=0   \61=1   \62=2   \63=3   \64=4   \65=5   \66=6   \67=7   \70=8   \71=9   \72=:   \73=;
\74=<   \75==   \76=>   \77=?   \100=≡  \101=A  \102=B  \103=X  \104=Δ  \105=E  \106=Φ  \107=Γ  \110=H  \111=I  \112=θ
\113=K  \114=Λ  \115=M  \116=N  \117=O  \120=Π  \121=Θ  \122=P  \123=Σ  \124=T  \125=Y  \126=ζ  \127=Ω  \130=Ξ  \131=Ψ
\132=Z  \133=[  \134=∴  \135=]  \136=⊥  \137=_  \140=—  \141=α  \142=β  \143=χ  \144=δ  \145=ε  \146=φ  \147=γ  \150=η
\151=ı  \152=φ  \153=κ  \154=λ  \155=μ  \156=ν  \157=ο  \160=π  \161=θ  \162=ρ  \163=σ  \164=τ  \165=υ  \166=ϖ  \167=ω
\170=ξ  \171=ψ  \172=ζ  \173={  \174=|  \175=}  \176=~  \177=  \200=  \201=  \202=  \203=  \204=  \205=  \206=
\207=  \210=  \211=  \212=  \213=  \214=  \215=  \216=  \217=  \220=  \221=  \222=  \223=  \224=  \225=
\226=  \227=  \230=  \231=  \232=  \233=  \234=  \235=  \236=  \237=  \240=€  \241=¥  \242=′  \243=≤  \244=÷
\245=∞  \246=f  \247=♣  \250=♦  \251=♥  \252=♠  \253=↔  \254=↵  \255=↑  \256=→  \257=↓  \260=°  \261=±  \262="   \263=≥
\264=×  \265=∞  \266=∂  \267=•  \270=÷  \271=≠  \272=≡  \273=≈  \274=...  \275=|  \276=—  \277=⌋  \300=⌘  \301=ℑ  \302=℔
\303=∅  \304=  \305=⊕  \306=∅  \307=∩  \310=∪  \311=⊃  \312=⊇  \313=∝  \314=⊂  \315=⊆  \316=∈  \317=∉  \320=∠  \321=∇
\322=®  \323=©  \324=™  \325=Π  \326=  \327=  \330=¬  \331=∧  \332=∨  \333=↔  \334=  \335=↑  \336=⇒  \337=  \340=∅
\341=∠  \342=®  \343=©  \344=  \345=Σ  \346=∫  \347=|  \350=∫  \351=∫  \352=∫  \353=∫  \354=∫  \355=∫  \356=∫  \357=∫
\360=  \361=∠  \362=∫  \363=∫  \364=∫  \365=∫  \366=  \367=∫  \370=∫  \371=∫  \372=∫  \373=∫  \374=  \375=∫  \376=∫
```