NUMERICAL RECIPES

Webnote No. 26, Rev. 2

Code for PSpage and PSplot

psplot.h

```
struct PSpage {
```
Object for generating a PostScript page.

```
    static FILE *PLT;
    static char *file;
    char fontname[128];
    Doub fontsize;

    PSpage(char *filnam) {
```
Constructor. Argument is the name of the PostScript file to be created.
```
        file = new char[128];
        strcpy(file,filnam);
        PLT = fopen(file,"wb");
        if (!PLT) throw("failure opening output file for plot");
        fprintf(PLT,"%%!\n/mt{moveto}def /lt{lineto}def /np{newpath}def\n");
        fprintf(PLT,"/st{stroke}def /cp{closepath}def /fi{fill}def\n");
        fprintf(PLT,"/zp {gsave /ZapfDingbats findfont exch ");
        fprintf(PLT,"scalefont setfont moveto show grestore} def\n");
        setfont("Times-Roman",12.);
        setlinewidth(0.5);
    }
    PSpage() {}
```
Alternative constructor, used internally to bind a `PSplot` to a `PSpage`.
```
    ~PSpage() {if (PLT) close();}

    void setfont(char *fontnam, Doub size) {
```
Change font from default (12 pt Times-Roman).
```
        strcpy(fontname,fontnam);
        fontsize = size;
        fprintf(PLT,"/%s findfont %g scalefont setfont\n",fontnam,size);
    }

    void setcolor(Int r, Int g, Int b) {
```
Change color from default (0,0,0=black). Range is 0 to 255.
```
        fprintf(PLT,"%g %g %g setrgbcolor\n",r/255.,g/255.,b/255.);}

    void setdash(char *patt, Int phase=0) {
```
Change line pattern from default (solid). Call with `""` to reset to solid.
```
        fprintf(PLT,"[%s] %d setdash\n",patt,phase);}

    void setlinewidth(Doub w) {fprintf(PLT,"%g setlinewidth\n",w);}
```
Change line width from default (0.5 pt).

```
    void setgray(Doub w) {fprintf(PLT,"%g setgray\n",w);}
```
Change gray level from default (0.0=black). Range is 0 to 1.

```
    void gsave() {fprintf(PLT,"gsave\n");}
```
Do a PostScript gsave.

```
    void grestore() {fprintf(PLT,"grestore\n");}
```
Do a PostScript grestore. Restores settings prior to last gsave.

```
    void rawps(char *text) {fprintf(PLT,"%s\n",text);}
```
Emit a string to the plot file as raw PostScript.

```
    void addtext(char *text) { fprintf(PLT,"(%s) show ",text); }
```
Plot text at current location.

```
    void puttext(char *text, Doub x, Doub y, Doub rot=0.0) {
```
Plot text at page location x, y (in pts) at angle rot.
```
        fprintf(PLT,"gsave %g %g translate %g rotate 0 0 mt ",x,y,rot);
        addtext(text);
        fprintf(PLT,"grestore \n");
    }
```

```
    void putctext(char *text, Doub x, Doub y, Doub rot=0.0) {
```
Plot centered text at page location x, y (in pts) at angle rot.
```
        fprintf(PLT,"gsave %g %g translate %g rotate 0 0 mt (%s) ",x,y,rot,text);
        fprintf(PLT,"dup stringwidth pop 2 div neg 0 rmoveto show grestore\n");
    }
```

```
    void putrtext(char *text, Doub x, Doub y, Doub rot=0.0) {
```
Plot right-justified text at page location x, y (in pts) at angle rot.
```
        fprintf(PLT,"gsave %g %g translate %g rotate 0 0 mt (%s) ",x,y,rot,text);
        fprintf(PLT,"dup stringwidth pop neg 0 rmoveto show grestore\n");
    }
```

```
    void close() {fprintf(PLT,"showpage\n"); fclose(PLT); PLT = NULL;}
```
Close the plot file. Called automatically by destructor.

```
    void display() {
```
Start external process to display the plot file.
```
        char cmd[128];
        if (PLT) close();
        strcpy(cmd,"\"C:\\Program Files\\Ghostgum\\gsview\\gsview32.exe\" ");
```
Change the above line to be your PostScript viewer.
```
        strcat(cmd,file);
        system(cmd);
    }
```

```
    void pointsymbol(Doub x, Doub y, Int num, double size) {
```
Plot Zapf Dingbat symbol num in page coordinates with specified size.
```
        fprintf(PLT,"(\\%03o) %g %g %g zp\n",num,x-0.394*size,y-0.343*size,size);
    }
```

```
    void lineseg(Doub xs, Doub ys, Doub xf, Doub yf) {
```
Draw a line segment in page coordinates (pts).
```
        fprintf(PLT,"np %g %g mt %g %g lt st\n",xs,ys,xf,yf);
    }
```

```
    void polyline(VecDoub &x, VecDoub &y, Int close=0, Int fill=0, Int clip=0) {
```
Draw connected line segments in page coordinates (pts), with options to close and/or fill the curve, or to set the curve as a clip area.
```
        Int i,n=MIN(x.size(),y.size());
        fprintf(PLT,"np %g %g mt\n",x[0],y[0]);
        for (i=1;i<n;i++) fprintf(PLT,"%g %g lt\n",x[i],y[i]);
        if (close || fill || clip) fprintf(PLT,"cp ");
        if (fill) fprintf(PLT,"fi\n");
        else if (clip) fprintf(PLT,"clip\n");
        else fprintf(PLT,"st\n");
    }
};
```

```
struct PSplot : PSpage {
```
Object that represents an x, y plot box on the page. Note that a PSplot object can call all the methods of its PSpage. It overloads many of these methods with versions taking x, y user coordinates instead of p, q page coordinates.

```
    Doub pll,qll,pur,qur;
    Doub xll,yll,xur,yur;
    VecDoub xbox,ybox;
    Doub majticsz,minticsz;

    PSplot(PSpage &page, Doub ppll, Doub ppur, Doub qqll, Doub qqur)
```
Constructor. Bind to page, with the specified p, q page coordinates (measured in pts) for lower-left and upper-right corners.
```
        : pll(ppll), qll(qqll), pur(ppur), qur(qqur),
        xll(ppll), yll(qqll), xur(ppur), yur(qqur), xbox(4), ybox(4),
        majticsz(8.), minticsz(4.) {
            strcpy(fontname,page.fontname);
            fontsize = page.fontsize;
            setlimits(xll,xur,yll,yur);
        }

    Doub p(Doub x) {return pll + (pur-pll)*(x-xll)/(xur-xll);}
    Doub q(Doub y) {return qll + (qur-qll)*(y-yll)/(yur-yll);}
```
Functions returning page coordinates p, q (in points) from user plot coordinates x, y.

```
    void setlimits(Doub xxll, Doub xxur, Doub yyll, Doub yyur) {
```
Set user x and y values for the lower-left and upper-right corners of the plot object. Always required.
```
        xbox[0] = xbox[3] = xll = xxll; ybox[0] = ybox[1] = yll = yyll;
        xbox[1] = xbox[2] = xur = xxur; ybox[2] = ybox[3] = yur = yyur;
    }

    void lineseg(Doub xs, Doub ys, Doub xf, Doub yf) {
```
Draw line segment using user coordinates.
```
        PSpage::lineseg(p(xs),q(ys),p(xf),q(yf));
    }

    void polyline(VecDoub &x, VecDoub &y, Int close=0, Int fill=0, Int clip=0) {
```
Draw connected line segments using user coordinates. See `PSpage::polyline` for meaning of the options.
```
        Int i;
        VecDoub xx(x), yy(y);
        for (i=0;i<x.size();i++) xx[i] = p(x[i]);
        for (i=0;i<y.size();i++) yy[i] = q(y[i]);
        PSpage::polyline(xx,yy,close,fill,clip);
    }

    void dot(Doub x, Doub y, Doub size=2.) {
```
Plot a filled circle at the specified location, by default small.
```
        PSpage::pointsymbol(p(x),q(y),108,size);
    }

    void pointsymbol(Doub x, Doub y, Int num, double size) {
```
Plot a Zapf Dingbat symbol at the specified location and size (pts).
```
        PSpage::pointsymbol(p(x),q(y),num,size);
    }

    void lineplot(VecDoub &x, VecDoub &y) {polyline(x,y);}
```
Plot a curve from `x` and `y` vectors of points.

```
    void frame() {polyline(xbox,ybox,1,0);}
```
Draw frame around this plot.

```
void clear() {gsave(); setgray(1.); polyline(xbox,ybox,1,1); grestore();}
```
Erase the interior of this plot.

```
void clip() {gsave(); polyline(xbox,ybox,1,0,1);}
```
Set interior of this plot as clip area.

```
void clip(VecDoub &x, VecDoub &y) {gsave(); polyline(x,y,1,0,1);}
```
Set clip area from `x` and `y` vectors of points.

```
void unclip() {grestore();}
```
Undo previous clip area (or anything else set subsequently).

```
void xlabel(char *text) {putctext(text,0.5*(pll+pur),qll-2.*fontsize-8.);}
```
Put text label on the x axis.

```
void ylabel(char *text) {putctext(text,pll-3.*fontsize-8.,0.5*(qll+qur),90.);}
```
Put text label on the y axis.

```
void label(char *text, double x, double y, double rot=0.) {puttext(text,p(x),q(y),rot);}
```
Put text label at an arbitrary position and rotation.

```
void scalestr(char *str, double x) {
```
Format a string for axis labels. Used internally.
```
    if (abs(x) < 1.e-15) x = 0.;
    sprintf(str,"%g",x);
}
```

```
void scales(Doub xmajd, Doub xmind, Doub ymajd, Doub ymind,
Int dox=2, Int doy=2, Int doxx=1, Int doyy=1) {
```
Draw scales (tick marks) on the plot. The x and y major and minor division intervals are specified by the first four arguments. The "do" arguments have values 0 (no ticks), 1 (ticks only), or 2 (ticks and numbers). `x` and `xx` are the bottom and top sides, `y` and `yy` are the left and right sides.
```
    char str[128];
    Doub x,y,xlo,ylo;
    if (dox || doxx) {
        xlo = ceil(MIN(xll,xur)/xmajd)*xmajd;
        for (x=xlo;x<=MAX(xll,xur);x+=xmajd) {
            scalestr(str,x);
            if (dox>1) putctext(str,p(x),qll-fontsize-2.);
            if (dox) PSpage::lineseg(p(x),qll,p(x),qll+majticsz);
            if (doxx) PSpage::lineseg(p(x),qur,p(x),qur-majticsz);
        }
        xlo = ceil(MIN(xll,xur)/xmind)*xmind;
        for (x=xlo;x<=MAX(xll,xur);x+=xmind) {
            if (dox) PSpage::lineseg(p(x),qll,p(x),qll+minticsz);
            if (doxx) PSpage::lineseg(p(x),qur,p(x),qur-minticsz);
        }
    }
    if (doy || doyy) {
        ylo = ceil(MIN(yll,yur)/ymajd)*ymajd;
        for (y=ylo;y<=MAX(yll,yur);y+=ymajd) {
            scalestr(str,y);
            if (doy>1) putrtext(str,pll-4.,q(y)-0.3*fontsize);
            if (doy) PSpage::lineseg(pll,q(y),pll+majticsz,q(y));
            if (doyy) PSpage::lineseg(pur,q(y),pur-majticsz,q(y));
        }
        ylo = ceil(MIN(yll,yur)/ymind)*ymind;
        for (y=ylo;y<=MAX(yll,yur);y+=ymind) {
            if (doy) PSpage::lineseg(pll,q(y),pll+minticsz,q(y));
            if (doyy) PSpage::lineseg(pur,q(y),pur-minticsz,q(y));
        }
    }
}
```

```
    void autoscales() {
```
Draw scales making reasonable default choices.
```
        double xmajd, xmind, ymajd, ymind;
        xmajd = pow(10.,((Int)(log10(abs(xur-xll))-1.1)));
        xmind = xmajd/5.;
        ymajd = pow(10.,((Int)(log10(abs(yur-yll))-1.1)));
        ymind = ymajd/5.;
        scales(xmajd,xmind,ymajd,ymind);
    }
};
FILE *PSpage::PLT;
char *PSpage::file;
```