Summary of ACCIS plotting system routines

I.H.Hutchinson

Contents

1 Apologia

The ACCIS plotting library is a lightweight portable library that gives fortran what it ought to have had: simple graphical output of scientific data. There are other libraries that fulfil that purpose; but this happens to be the one I wrote; so I like it. Also it is entirely free and open, so it can be included in the distribution of any program, without encumbrance.

It is powerful. For example, it has interactive routines for displaying selectable slices of 3-D data, rotating the views of surfaces, projecting 2-D contours, and producing smoothly coloured contours.

It also produces publication-quality vector-drawn postscript output, including text and mathematical symbols.

It observes what ought to be a standard principle of scientific programming: that the focus should be the calculation, not the graphical interface. In other words, the fortran program calls the graphical display, rather than what has become the standard approach in windowing systems that the graphical interface calls the calculation. The result is that the learning curve is easy, and one can get started with a couple of subroutine calls.

Finally, it is written in fortran 77 (with the exception of the X or Win32 graphics driver which need C) so if you can compile your fortran program, you can compile the accis routines.

The best way to get into using accis is by example. So I recommend looking at the routine plottest.f, and others in the directory testing.

2 Conventions

For purposes of this summary, the following implicitly indicate types: i: integer, c: character, l: logical, all others: real. Terminating letter indicates: v: vector (1-d array), a: 2-d array.

There are world units, normalized (normal) units, and box units, each has a direct scaling to the others. Users can plot in world units standard values. Normalized x-units are 0 to 1 across the width of the canvas and normal y-units have the same size as x-units and start at 0 at the bottom, but the top of a landscape canvas doesn't reach 1. Box units are fractional distance 0-1 across the plot-region box, in either direction.

3 Automatic plotting, single call plots.

AUTOPLOT(xv,yv,ilength) Line plot yv versus xv; linear axes.

AUTOMARK(xv,yv,ilength,isym) Plot using symbol isym.

LAUTOPLOT(xv,yv,ilength,lnx,lny) Line plot with logarithmic axis in x or y or both.

LAUTOMARK(xv,yv,ilength,lnx,lny,isym) Symbol plot with logarithmic axis(es).

YAUTOPLOT(yv,ilength) Line plot yv versus index; linear axes.

4 Setup

PLTINIT(xmin,xmax,ymin,ymax) Switch to graphics, set world scaling.

FITINIT(xmin,xmax,ymin,ymax) same but fitting nice ranges.

ACCISINIT() Shorthand call for PLTINIT(0.,1.,0.,1.).

AUTOINIT(xv,yv,ilength) Set up scaling and axes as for AUTOPLOT based on the data, but don't actually do the plotting. This is useful if one wishes e.g. to change the color of the line before it is plotted by a call to POLYLINE.

LAUTOINIT(xv,yv,ilength,lnx,lny) Set up possibly logarithmic scaling and axes, without drawing.

SCALEWN(xmin,xmax,ymin,ymax,lnx,lny) Set world scaling explicitly, possibly log.

FITSCALE(xmin,xmax,ymin,ymax,lnx,lny) same but fitting nice ranges.

PLTINASPECT(xmin,xmax,ymin,ymax) Initiate graphics. Resize axis region to retain aspect ratio of plot with x,y in the same units. Set world scaling.

AXREGION(xin,xan,yin,yan) Set the axis region in normal units. Default: .31,.91,.1,.7

AXPTSET(xfrac,yfrac) Set the primary axis point (i.e. intersection) at specified fraction of the plot region (i.e. in box units). (Default 0.,0.)

MULTIFRAME(irows,icolumns,ispace) For subsequent plots (using pltinit), plot irows x icolumns plots per page, with label space determined by ispace (bit0 implies space x, bit1 y). Call with irows=0, single frame.

PFSET(inum) Set file plotting for hardcopy. inum=(0:no, 1:hp, 2:ps, 3:eps). If inum.eq.-1, then prompt for choice. If inum.lt.-1 output to file only, not screen.

PFPSSET(ips) Set the use of postscript character fonts (ips=1) or vector drawn fonts (0, default), or (3) use the value previous to current. Can be turned on and off during the plot. Has no effect on screen display, only on postscript output.

IPFPS() Return value of PFPS setting.

WINSET(lwin) Set windowing to axis-region on or off (.false. default).

TRUNCF(xmin,xmax,ymin,ymax) Set rectangular truncation (windowing) norm-units. If all args=0, switch off (default).

PLTEND() Flush graphics, wait for mouse click in plotting window; then return.

5 Data plotting

POLYLINE(xv,yv,ilength) Draw a polyline on an already setup graph.

YPOLYLINE(yv,ilength) Draw a polyline on an already setup yautoplot.

POLYMARK(xv,yv,ilength,isym) Draw data marked by symbols isym: 1 hollow square, 2 hollow triangle, 3 bullet, 4 solid square, 5 solid triangle, 6 hollow star, 7 solid star, 8 hollow diamond, 9 flag, 10 +, 11 x, 12 diamond, 13 -, 14 inverted triangle, 15 circle. Else char(isym), so ichar('c') gives 'c', ichar('c')+128 gives the font-2 symbol corresponding to 'c' (xi).

DASHSET(idash) Set dashed line character for polyline. (0:solid, 1:long, 2:short, 3:long/short,

4:dots)

LABELINE(xv,yv,ilength,clabel,ilablen) Draw polyline of ilength points, with embedded label, clabel, of length ilablen characters. If ilablen.eq.-99 then set the normalized interval between labels to xv(1); if xv(1).eq.0 set it to default 0.3.

POLYDRAW(xv,yv,ilength,DRAWFN) Draw data marked by symbol defined by the routine DRAWFN, a subroutine accepting 2 real arguments, being the normalized x and y position of the symbol center, which draws the symbol. Specify EXTERNAL DRAWFN in the calling routine. Built-in routines are ACCIRCLE, ACTRID, ACAST, ACX, ACPLUS.

ACGEN(x,y) is a general routine to be passed to POLYDRAW that will draw a user-defined symbol without having to define an actual function. Instead the subroutine

ACGSET(xv,yv,ilength,ifill) is called first with arguments giving the symbol vertices in units of the charsize, and calling for filling of the symbol if ifill .ne. 0. Thereafter acgen generates the symbol specified. Since the width and height are scaled by chrswdth and chrshght, aspect-ratio can be scaled using charsize().

POLYERRS(xv,yv,errv,ilength,u,d) Draw error bars from yv-d*errv to yv+u*errv for all the xv, yv, errv arrays.

POLYBOX(xv,yv,ilength) Draw the outline of a histogram of boxes whose (touching) boundaries are at xv(0:ilength) and whose heights are yv(ilength). Ends have y=0. Notice that there is one more xv than yv!

STPOLYLINE(xa,ya,ilength,ixstride,iystride),

STPOLYMARK(xa,ya,ilength,ixstride,iystride,isym) Strided versions of polyline/mark, in which a total of ilength data points are sampled from the arrays xa and ya at strides of ixstride and iystride. These allow one to plot data from rows (rather than columns) of a matrix by passing the leading dimension of the matrix as the stride. Passing strides of 1 gives sampling equivalent to the unstrided versions.

6 Color

COLOR(icol) Set current discrete color of 0-15. Default 15 black (on white). If icol>15, then instead call GRADCOLOR(icol-15).

GRADCOLOR(icol) Set current color from a 1-240 color gradient range.

ACCISGRADINIT(ir1,ig1,ib1,ir2,ig2,ib2) Set the gradient linear color range start and end, rgb amplitudes 0-65535.

ACCISGRADSET(iredv,igreenv,ibluev,npixel) Set the color gradient with arbitrary RGB values from vectors of length npixel.

IGETCOLOR() Integer function. Get the number of the current color. [0-15 discrete, 16-256 a gradient color].

PATHFILL() Fill the path just drawn, with the current color. A path is a set of vectors all drawn with pen down one after the other.

IDARKBLUE(), IDARKGREEN(), ISKYBLUE(), IBRICKRED(), IPURPLE(), IGOLD(), IGRAY(), ILIGHTGRAY(), IBLUE(), IGREEN(), ICYAN(), IRED(), IMAGENTA(), IYEL-

LOW(), IBLACK(): Integer functions returning the discrete 16-color number that gives what their name indicates ($= 1 - 15$).

GRADTRI(xv,yv,zv,dv,zg0,zg1,ng0,ng1,isw) shade triangle specified by vertices xv(3), yv(3)[, zv(3)], with gradient values dv(3). The lower/upper parameter values zg0,1 are mapped to gradient colors ng0,1. isw bit 0 sets 2D (0) or 3D (1); bit 1 indicates use of dv: either absolute parameter values (0), or as direction cosines (1), in which case value is the point (xv,yv,zv) coordinate in that direction.

GRADQUAD(xqv,yqv,zqv,dqv,zg0,zg1,ig0,ig1,isw) fill a quadrilateral by splitting into four triangles at the centroid and using GRADTRI.

GRADLEGEND(c1,c2,x1,y1,x2,y2,colwidth,lpara) draw a color gradient axis-legend. Value corresponding to color goes from c1 to c2: the ends of the color gradient. Position of axis is (x1,y1) to (x1,y2) in axis-box units. Width of color bar is colwith times axis-box width. lpara=.true. says use parallel labels.

agradcolor(icol) set gradient color 0-239.

getrgbcolor(icol,ired,igreen,iblue) get the gradient color RGB settings for input number icol 0-239.

7 Axes

AXIS() Automatic x and y axis.

AXIS2() Put tics on the opposite side of the axis box.

XAXIS(first,delta) Draw an axis; labels starting at first, spaced by delta. If delta=0, autofit labels, which is what AXIS() does for both axes.

YAXIS(first,delta) If axis is log, first indicates major subtic.

ALTXAXIS(xi,xa) Draw an alternative xaxis whose ends have world values scaled by the factors xi and xa.

ALTYAXIS(yi,ya) Normally the axis position would be moved by axptset(1.,1.) and probably ticrev(), all manually, before drawing.

AXLABELS(cxlab,cylab) Label the x/y axes with strings.

BOXTITLE(ctitle) Label the plot with a string centered at the top.

GAXIS(amin,amax, igpow,first,delta, xmin,xmax,ymin,ymax, lpara,laxlog) General axis with ends positioned at (xmin, ymin) (xmax, ymax) normal units. World axis labels based on world end values amin, amax, first, delta, igpow (power of ten label). Lpara puts labels parallel (else dperp). Laxlog logarithmic.

AXPTSET(xpt,ypt) Set the axis intersection point at specified fractions of the full lengths of the axes. This determines where the axes are drawn.

TICSET(xlen,ylen,xoff,yoff,ixw,iyw,ixp,iyp) Set tic lengths, label offsets, widths, decimal points. Labels switched off by width ixw$\leq$0. Defaults set respectively if reals all =0 or ints all =0. (Defaults: .015,.015,-.03,-.02,4,4,1,1)

TICREV() Reverse the tics and labels to the other side of axis.

TICLABTOG() Toggle the tic labels on and off.
TICNUMSET(itics) Set the number of tics of axis (approximately).

8 Text plotting

Special text is accessed via "!" followed by one of the following:

```
@       to normal font 0.
A       to font 1 (math)
B       to font 2 (italic)
R       to font 3 (roman)
E       to font 4 (english gothic)
G       to font 5 (german gothic)
d/D     Toggle subscript mode
u/U     Toggle superscript mode
p       Save this position
q       Save current position and return to previously saved position.
o       Toggle Write-over mode
n/N     Toggle Write-under mode
```

JDRWSTR(xn,yn,cstring,just) Draw cstring at (xn,yn)(normal), justified per parameter just: 0. centered, +1. normal left-justified, etc.

DRCSTR(cstring) Draw cstring from current position. Leave at end of string.

DRWSTR(xn,yn,cstring) Draw cstring from (xn,yn)norm. Leave at end of string.

WSTR(cstring) Real function: normal-units length of string at current size.

CHARSIZE(width,height) Set character size. (0,0) sets default (.015,.015).

CHARANGL(theta) Set character angle to horizontal in degrees.

GETCANGL(theta) Get character angle to horizontal in degrees.

ANNOTE [DOS only] Enter interactive annotation mode. Type help for help.

GETFONT(fontname) Read a new fontset from file.

LEGENDLINE(xg,yg,nsym,cstring) Draw a legendline at fractional position in plot box: xg, yg. Use symbol nsym if positive and $\leq$256. If nsym<0 use both line and symbol. If nsym=0 put only line. If nsym=257 use nothing but put the string at the usual place. If nsym=258 use nothing but put the string at the start of the line.

9 Contouring

CONTREC(za,cworka,ixm,iym,zclv,icl) Simple contour of entire array za(ixm, iym) at levels zclv(icl). If icl=0 fit contour levels instead. If icl.lt.0 no line labels. Cworka is a work character array at least (ixm, iym).

CONTOURL(za,cworka,iLdim,ixm,iym,zclv,icl,x,y,icsw) General contour za(1:ixm(of ildim), iym) at zclv(icl) (or fit if icl=0, using zclv(1) if non-zero to determine contour number) on a mesh defined by x,y. Switch icsw determines call type: 0 equal spacing, x,y not used; 1 vectors x,y determine mesh; 2 arrays x,y determine mesh; bit 5 (16) sets coloring; bit 6 (32) omits contour lines; bit 7 (64) color-fills with triangle gradients, in which case non-zero second byte determines the step size of the gradients (1 by default). Color contouring should be done with icl .ge. 2 and the maximum and minimum values in zclv(1) and zclv(icl). Then a gradlegend can be constructed with these values.

CONT3PROJ(za,cworka,iLdim,ixm,iym,zclv,icl,x,y,icsw,f) Draw a contour plot projected on to the z=f*scbz3 (constant) plane of a 3-D plot. Axes etc can be added manually. To end projection call hdprset(0,0.). All the arguments but the last are those of CONTOURL.

CONTOUR(za,xa,ya,ix,iy,zclv,icl) Simple contour over mesh xa,ya of entire arrays, needing no extra work space. No line labelling.

AUTOCOLCONT(za,iLdim,ixm,iym) Color contour za on rectangular mesh.

ARROWPLOT(E1,E2,Erange,Li,imax,jmax,x,y,iasw) Plot arrows to represent vector whose x-,y-components are E1(imax(or Li),jmax),E2(imax,jmax), defined on the position vectors/arrays represented by x,y. iasw works like icsw (see CONTOURL). Arrow length is 1 normalized unit times E/Erange. Plot region should be setup up prior to call.

10 Projected 3-D surface routines

HIDWEB(xv,yv,za,iL,ix,iy,isw) Draw a 3-D web of za(ix,iy) (leading dimension iL) at xv,yv. View obtained from file eye.dat (3 reals) or default if nonexistent. Control switch isw: .lt.0 no axes. Lowest byte abs(isw)=

```
0 don't rescale, use last scaling and perspective
1 scale to fit 1-D range xv(1)-xv(ix), yv(1)-yv(iy), zmin-zmax.
2 scale to 2-D xmin,xmax, ...; don't hide lines, just wiremesh.
Bit3 set [4]. Do no perspective setting regardless.
```

Higher bytes: 2nd (*256) color of web if non-zero. 3rd (*65536) color of axes (if !=0). (Remains set).

SURF3D(xa,ya,za,iL,ix,iy,isw,worka) Draw an opaque 3-d surface of za(ix,iy), (leading dimension iL) at xa(ix,iy),ya(ix,iy). View obtained from file eye.dat (3 reals) or default if nonexistent. Switch isw: .lt.0 no axes; lowest nibble =0 or 4 use last scale, if =0 xa, ya must be 2-D arrays; =1 scale to fit x,y (1-D) and xa,ya are vectors, =2 scale to fit xa,ya (2-D arrays). Second byte (*256) color of web if non-zero. Third byte (*65536) color of axes (if !=0). (Remains set). Second nibble (16+) is isw for the SURFDR3 call. Array worka(0:iL+1,0:iy+1) must be provided. Surface is normally (isw=0) drawn as quadrilaterals filled with gradient-color according to (za) height scaled to the total height. This is the higher level routine with built in scaling etc., similar to hidweb but with shaded surfaces.

SURFDR3(xa,ya,za,iL,ix,iy,worka,isw[,dv]) Draw 3-D surface using current projection. Switch isw bit 0 set: triangular fills, else chunks. If, in addition, isw bit 1: directional shading using

optional argument dv(5) dv(1-3) gives direction dv(4-5) gives distance limits, else z-height shading. Use for drawing closed parameterized surfaces.

AXON(xv,yv,za,iL,ix,iy) Plot axonometrically za(ix,iy) at xv,yv (i.e. staggered slices at constant y, no perspective). Use eye.dat data, if exists; first two values indicate normal distance offset of axis end.

WEBDRW(xv,yv,za,iL,ix,iy,icorner) Draw a 3-D web using current projection. Return closest corner to view point in icorner.

HIDINIT(top,bot) Set the top and bottom hiding horizons (usually 0.,1.).

HDPRSET(isw,fxd) Set hiding and 2-3 projection per switch isw. Zero off. Positive: hiding only. Negative: hiding and projection with fixed coordinate either (x,y,or z) for isw=(-1,-2 or -3) having value fxd. For left handed system, isw=(-4,-5 or -6).

HDPROJECT(ihide,iproject,ifixedcoord,fxdworld,irlsys) Set hiding (ihide = 1), projecting 2→3 (iproject.ne.0), which coordinate to hold fixed (ifixedcoord), what its world value is (fxdworld), and right (+1) or left (-1) handed system (irlsys).

11 Three-Dimensional Perspective Drawing

POLY3LINE(xv,yv,zv,ilength) Draw a polyline in 3-D. (Minimum setup call prior to this is just pltinit.)

SCALE3(xmin,xmax,ymin,ymax,zmin,zmax) Set 3-D world scaling to cube size. Defaults min=-1, max=+1.

SETCUBE(xc,yc,zc,xcen,ycen) Set norm-value of cube edges (+-) and 2-d position of cube center. Defaults .25,.25,.2,.5,.4.

TRN32(x,y,z,xt,yt,zt,ifl) If ifl=1, set 3-D transform; (x,y,z) is point looked at, (xt,yt,zt) is eye. If ifl=2, set axonometric, offset xt,zt. If ifl=0 transform (x,y,z) to (xt,yt). If ifl=-1 return eye position in xt,yt,zt.

CUBEPROJ() Draw the cube outline faces closest to eye.

IGETCORNER() Integer function returning the closest corner to eye.

AXPROJ(ic) Draw axes, centered at corner given by ic. Corner=bits0-3(ic). If bit4(ic)=1, flip labels. If bit5(ic)=1, x-labels vertical; if bit6(ic)=1, y-labels vertical, else horizontal. Normally AXPROJ(IGETCORNER()).

CUBED(icorner) Draw a cube outline when icorner is the nearest. Code is 1,2,3,4 anticlockwise (from top) from x3min,y3min, + top - bottom.

AXIDENT3() Identify the 3 axes (logically x, y, z).

AX3LABELS(charx,chary,charz) Label the three 3-D axes.

HIDVECN(x2,y2,ipen) Draw a (2-D) vector, hiding appropriately.

SCBN(icoord) Real function returns normalized value position of cube face corresponding to coordinate icoord=constant (icoord =1,2,3). So for example to project onto the z=constant (negative) face of the cube, you do: call hdprset(-3,-scbn(3))

TN2S(px,py,sx,sy) Transform normalized px,py to screen coordinates sx,sy. If (world3.h)

ihiding is < 0, project in direction of 3-d axis mod(abs(ihiding),2) at value fixedn, right/left-handed when $(1 \leq -ihiding \leq 3)$ or $(4 \leq -ihiding \leq 6)$.

EYE3D(isw) Enter interactive rotation of the view of 3-D drawing. The routine responds to a mouse drag in the window. On button release the routine exits with isw=1 if motion occured or isw=0 if no motion. Normally motion calls for a redraw of the plot and reentering eye3d, while no motion is the signal to continue. Thus the following code gives interactive examination of the object:

```
51      Continue
        Drawing commands ...
        call eye3d(isw)
        if(isw.ne.0)goto 51
```

Key presses with focus in the plot window, return isw= keysym of key pressed. For example in X, left, up, right, down cursor arrows are ff51-4 respectively (65361-4 decimal). Letters are their lower case ASCII codes, etc. The return value can be used for other control purposes. But the codes for cursors are different under Windows (37-40) and it returns the ASCII codes for upper case.

IEYE3D() Integer function of convenience to return switch value of eye3d.

ROTATEZOOM(isw) Called immediately after eye3d sets isw, this routine adjusts the perspective by zooming-view (z,x), rotating-view (e,r), or moving in/out (i,o). The drawing code that is then repeated ought not to set the perspective explicitly (through a call to trn32) else a call to puteye must be inserted after rotatezoom to ensure the eye.dat is updated, otherwise the adjustment will be overridden.

DRAW3ARROW(arrowdata) Draw a 3-D arrow according to settings of arrowdata(10): Base position world coords (3), Point position (3), Barb parallel, perp size as fraction of length (2). Feather size parallel, perp, frac of length (2). If barbperp arrow(8) is negative, then arrowplot must have total length 12, and the last two parameters control (11) the number of barb angles and (12) whether the barb is filled or not.

ARG3ARROW(a1,a2,a3,a4,a5,a6,a7,a8,a9,a10[,a11,a12]) call DRAW3ARROW with explicitly passed individual arguments.

12 Plotting volumetric 3-D Data

SLICEGWEB(ifullv,iudsv,u,nw,zp,ixnpv,xnv,ifix,cutitle,svec,vp) Plot as 3-D webs slices through 3-D array u. ifullv(3) and iudsv(3) are the full and used dimensions of u. zp(nw,nw) is a work array; nw should be >max(iudsv). ixnpv(3) are the pointers within xnv to the node position array starts. That is, the positions of the nodes in dimension id are stored in xnv(ixnpv(id))+1 to xnv(ixnpv(id))+ iudsv(id). ifix is the initial fixed dimension (1,2 or 3). cutitle is the character array title of u. Keyboard control of the slice is provided, together with perspective view controlled by mouse. If the third bit of abs(ifix) is set (by adding 4 to it), then overplot on the contour plot arrows representing the projection of the vector field svec(ifullv,3), which has the same indexing and positions as u, and three vector components. The work array vp(nw,nw,2) must be provided. The last two arguments may be dummy in

the call if abs(ifix)<4.

SLICEGCONT(ifullv,iudsv,u,nw,zp,ixnpv,xnv,ifixptv,cutitle,svec,vp) Plot perspective view of slices on which are contours of 3-D array u(ifullv(1),ifullv(2),ifullv(3)). ifullv(3), iudsv(3), zp(nw,nw), ixnpv(3), cutitle are as above in slicegweb. The initial intersection of the 3 fixed planes is at ifixptv(3). Slightly different keyboard/mouse control is provided. Overplotting arrows is indicated by the third bit of ifix being set; the work array vp(nw,nw,3,3) must then be provided.

13 Primitives

VECW(x,y,ipen) Draw world-unit vector to (x,y) with pen up (ipen=0) or down (1). If ipen=-1, then just put a point at (x,y).

VECN(x,y,ipen) Draw norm-unit vector to (x,y) with pen up (0) or down (1).

WX2NX(wx), WY2NY(wy) Real functions convert world to normal units.

XN2XW(wx), YN2YW(wy) Real functions convert normal to world units.

VEC3W(x,y,z,ipen) Draw world vector to (x,y,z).

VEC3N(x,y,z,ipen) Draw norm vector to (x,y,z). Domain (-1.,1.) covers the cube.

WXYZ2NXYZ(xw,yw,zw,xn,yn,zn) Transform from world 3-D vector to normalized.

GTIC(xgw,ilab,xg,yg,tcos,tsin,axcos,axsin,lpara) Low level tic drawing.

GETEYE(x,y,z) Return the position of view in 3-D perspective plots.

PUTEYE(x,y,z) Set the position of view.

14 Utilities

FITRANGE(min,max,itics,ipow,fac10,delta,first,xlast) Fit a maximum of itics ticks to range (min, max). Decide a scaling factor 10^{ipow}=fac10 and a sensible (scaled) delta between tics. First (xlast) is integer multiple of delta lying closest to min (max) outside the range (min,max). So fac10*(first+i*delta) are the tick values.

MINMAX(xv,isize,min,max) Find min and max of vector of size isize.

MINMAX2(xa,iL1,ix,iy,min,max) Find min and max of 2-d array of leading dimension iL1, over indices (ix,iy).

FWRITE(x,iwdth,ipoint,cstring) Format write the float x into cstring with specified ipoint decimal places. Total width returned as iwdth.

IWRITE(i,iwdth,cstring) Format the integer i into cstring, return iwdth.

TERMCHAR(cstring) Terminate the string in 0, truncating trailing spaces.

ACCISFLUSH() Flush all graphics to the display before proceeding. This is useful if the program is about to enter a polling state, waiting for input. It ensures that all the prior display of graphics is visible on entering that polling state.

ASLEEP(iusec) Insert a delay of approximately iusec microsec into both live plotting and

ps files, for making sequential drawing animations. Timing will be much slower over remote Xservers and serious bandwidth wasted because delays are generated by writing to the screen. When playing such ps files using ghostscript, the duration can be changed by an arbitrary (real) factor by predefining the gs macro SF using a command such as

gs -c /SF 2.5 def -f plot0001.ps

which causes the animation duration to be multiplied by 2.5

15 Linking

The routines are normally compiled and gathered into a library `libaccis...` which is subsequently linked against. Compiling is accomplished by issuing the simple command `$ make`. The makefile searches for a fortran compiler and builds the libary if successful. If it can't find one, then e.g. `$ make G77=path90` tells it the name of the compiler is `path90`. There are various other `make` targets that can be found in the Makefile.

The graphical output driver routines are contained in vecx.c, vecglx.c, vecwin.c, which require a C compiler, or the obsolete but occasionally useful vec4014.f. Of these driver files, the routines svga, txtmode, vec, vecfill, scolor, accisgraddef, acgradcolor, getrgbcolor are public, for calling by higher level routines, and possibly the user.

For X window systems, the X11 driver (vecx.c) requires linking against the Xlib library using compiler/linker arguments such as -L/usr/X11R6/lib/-lX11. The OpenGL driver vecglx.c requires in addition -lGL and -lGLU. The development versions of the X11 and OpenGL packages may be required to provide these.

All the ACCIS code with the exception of the drivers is fortran 77. It takes advantage of the ubiquitous essential extensions: long variable names and include statements, that make F77 still viable today. By virtue of fortran policy, it is therefore also fully compliant with the 1990, 1995, 2003, 2008 etc., fortran standards, without modification. However, when it is being called from modern (post 77) fortran, it may be advantageous for debugging purposes to use explicit interface definitions in the calling program. The file `interface.f90` will be generated by `$ make interface.f90` and can then be included in fortran code compiled with a F90 (or later) compiler. The explicit interface helps the compiler detect erroneous argument parameters to subroutine calls. It is not necessary for operation of debugged correct code.