Subject: Re: Spherical Contour Plotting
Posted by k-bowman on Thu, 10 Jan 2002 16:59:52 GMT
View Forum Message <> Reply to Message

In article <MPG.16a6689c231416089897c5@news.frii.com>, david@dfanning.com wrote:

> Kenneth Bowman (k-bowman@null.com) writes:
>
>> If the finished "thing" (I avoid the word object) is to be 2-D, try the
>> SATELLITE projection in MAP_SET.
>>
>> If you want a 3-D "thing", you could do a CYLINDRICAL EQUIDISTANT plot
>> with MAP_SET. CONTOUR will return the coordinates of the isopleths
>> (PATH_INFO, PATH_XY). Then transform the longitude-latitude coordinates
>> into 3-D space and use direct or object graphics to display the resulting
>> lines.
>
> Somehow I knew the answer to this was going to be easy. :^)

Oh, its not nearly as hard as object graphics. Here is a direct graphics
example.
Note that the work of extracting the contour lines and converting to
Cartesian coordinates is only a few lines of IDL code.

I leave it to you object wizards to turn the 3-D lines into objects and stick
a sphere object inside. ;-)

Ken

```
PRO SPHERICAL_PLOT

COMPILE_OPT IDL2

cr = ''

nx =
65
;Grid resolution in longitude
ny =
33
;Grid resolution in latitude

x  =
(360.0/(nx-1))*FINDGEN(nx)
;Longitude grid
y  = -90 +
(180.0/(ny-1))*FINDGEN(ny)
```

```
;Latitude grid
xx = x               # REPLICATE(1.0,
ny)                          ;2-D longitude grid
yy = REPLICATE(1.0, nx) #
y                                ;2-D latitude grid
z  = SIN(!DTOR*xx) *
COS(!DTOR*yy)                          ;Test function to
contour

;Standard contour plot on satellite map projection
MAP_SET, /SATELLITE, /CONT, /ISOTROPIC
CONTOUR, z, x, y, LEVELS = -0.95 + 0.1*FINDGEN(20),
/OVERPLOT            ;Contour z
PRINT, 'Enter <cr> to continue.'
READ, cr

;Get contour info
CONTOUR, z, x, y, PATH_INFO = path_info, PATH_XY = path_xy,
$            ;Contour z, save contour info
   /PATH_DATA_COORDS, CLOSED = 0, LEVELS =-0.95 + 0.1*FINDGEN(20)

;2-D plot using contour info
PLOT, [0, 0], [1, 1], /NODATA, $
   XTITLE = 'Longitude', $
   XSTYLE = 1, $
   XRANGE = [0.0, 360.0], $
   XTICKS = 4, $
   YTITLE = 'Latitude', $
   YSTYLE = 1, $
   YRANGE = [-90., 90.0], $
   YTICKS = 6

FOR k = 0, N_ELEMENTS(path_info)-1 DO BEGIN
   i0 =
path_info[k].offset
;First element of the k'th contour
   i1 = i0 + path_info[k].n -
1                              ;Last element of the k'th
contour
;  PRINT, k, path_info[k].type, i0, i1
   xc =
REFORM(path_xy[0,i0:i1])
;Extract x-coords of k'th contour
   yc =
REFORM(path_xy[1,i0:i1])
;Extract y-coords of k'th contour
   IF (path_info[k].type EQ 1) THEN
BEGIN                          ;Close contours, if needed
```

```
      xc = [xc, path_xy[0,i0]]
      yc = [yc, path_xy[1,i0]]
   ENDIF

   PLOTS, xc,
yc                                          ;Plot
contours
ENDFOR
PRINT, 'Enter <cr> to continue.'
READ, cr


;3-D plot using contour info
PLOT_3DBOX, [0,0], [0,0], [0,0], /NODATA, $
   XTITLE = 'X', $
   XSTYLE = 1, $
   XRANGE = [-1.0, 1.0], $
   XTICKS = 4, $
   YTITLE = 'Y', $
   YSTYLE = 1, $
   YRANGE = [-1., 1.0], $
   YTICKS = 4, $
   ZTITLE = 'Z', $
   ZSTYLE = 1, $
   ZRANGE = [-1.0, 1.0], $
   ZTICKS = 4

r = 0.9
FOR k = 0, N_ELEMENTS(path_info)-1 DO BEGIN
   i0 =
path_info[k].offset
;First element of the k'th contour
   i1 = i0 + path_info[k].n -
1                                   ;Last element of the k'th
contour
;  PRINT, k, path_info[k].type, i0, i1
   xc =
REFORM(path_xy[0,i0:i1])
;Extract x-coords of k'th contour
   yc =
REFORM(path_xy[1,i0:i1])
;Extract y-coords of k'th contour
   IF (path_info[k].type EQ 1) THEN
BEGIN                               ;Close contours, if needed
      xc = [xc, path_xy[0,i0]]
      yc = [yc, path_xy[1,i0]]
   ENDIF
```

```
   x3 = r * COS(!DTOR*yc) *
COS(!DTOR*xc)                          ;Convert to Cartesian
coordinates
   y3 = r * COS(!DTOR*yc) * SIN(!DTOR*xc)
   z3 = r * SIN(!DTOR*yc)
   PLOTS, x3, y3, z3, /T3D
ENDFOR

END
```