Subject: Re: GridData Conundrum
Posted by greg.addr on Sun, 18 Apr 2010 14:15:39 GMT
View Forum Message <> Reply to Message

On Apr 17, 7:19 pm, David Fanning <n...@dfanning.com> wrote:
> Folks,
>
> I have long thought that the IDL gridding routine, GridData,
> to be one of IDL's most powerful and useful routines. Perhaps
> taking its place among the likes of Histogram and Value_Locate.
> Well, it *would* be powerful and useful if I could ever
> get the damn thing to work. But, alas, I never have been
> able to accomplish this simple feat.
>
> I've decided to come clean about my abysmal failure
> and ask for your help.
>
> I ran into the perfect test case this week. A simple nearest
> neighbor gridding problem that I know how to solve in two
> completely independent ways, each producing identical
> results. I *know* what I am doing here and I am
> *supremely* confident in the results. "And," I thought,
> "it is so simple, I could do this in GridData!"
>
> Not. :-(
>
> I've explained the problem and put some data here on
> my web page:
>
>   http://www.dfanning.com/code_tips/usegriddata.html
>
> I would be *extremely* grateful to anyone who can take
> me by the hand and lead me to the promised land.
>
> Cheers,
>
> David
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming:http://www.dfanning.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Hi David,

I've been trying something very similar recently. I think the
confusion is with the map_project operation - what you want is to
convert the UV coordinates of the stereo projection into lat,lon

values, and not the lat,lon into equirectangular UV. In the end, though, you don't need Griddata gor it - hope that's not a disappointment!

You can recreate the map you want (http://hrscview.fu-berlin.de/mex4/software/other/out.png - sorry about the shocking colour!) like this:

```
pro tmp_fanning_map
  im=fltarr(144,73)
  openr,1,"D:\mydocs\work\2010-04-18 fanning\usegriddata.dat"
  readu,1,im
  close,1
  im=reverse(im,2)

  lat=gm_scl(indgen(73),out_range=[-90.,90.])
  lon=gm_scl(indgen(144),out_range=[0.,360])

  map=map_proj_init('Stereographic', center_lon=-45, center_lat=90,
sphere_radius=6378273.00)

  sz=[304,448]

  xr=[-385,375]*1e4
  yr=[-535,585]*1e4

  x=gm_scl(indgen(sz[0]),out_range=xr)
  y=gm_scl(indgen(sz[1]),out_range=yr)

  q=lindgen(product(sz))
  qx=q mod sz[0]
  qy=q/sz[0]

  lonlat=map_proj_inverse(x[qx],y[qy],map_structure=map)

  lon0=reform(lonlat[0,*])
  lat0=reform(lonlat[1,*])

  x0=wrap360(lon0)/360.*144
  y0=(lat0[q]+90.)/180.*73.

  out=fltarr(sz)
  out[q]=im[x0[q],y0[q]]

  device,decomposed=0
  loadct,11

  tvscl,out
end
```

You'll need these, too:

```
function gm_scl,x,in_range=in_range,out_range=out_range
  ;more powerful bytscl function - type taken from out_range if
present, otherwise x
  ;in_range - range of input data to be stretched
  ;out_range - output range

  tname=size(keyword_set(out_range)?out_range:x,/tname)
  type=size(keyword_set(out_range)?out_range:x,/type)

  mn=min(x,max=mx)
  if keyword_set(in_range) then begin
      mn=in_range[0]
      mx=in_range[1]
  endif

  if ~keyword_set(out_range) then begin
    eps=1d-9
    case type of
        "UINT":out_range=[0,65536-eps]
        "INT":out_range=[-32768,32768-eps]
        "BYTE":out_range=[0,256-eps]
        else:out_range=[0,100-eps]
    endcase
  endif

  y=(double(x)-mn)/(mx-mn)
  y=y>0d<1d
  out=out_range[0]+y*(out_range[1]-out_range[0])

  return,fix(out,type=type)
end

function wrap360,a ;make -179 and +179 close neighbours
 return,(a+360) mod 360
end
```