Subject: Re: Indices ?
Posted by tam on Thu, 02 Dec 2004 18:54:04 GMT
View Forum Message <> Reply to Message

sdj@tiscali.it wrote:

> OK, so let me explain what I need to do and why I think that the
> aforementioned information is necessary. Please excuse me if this
> message is a tad long...
>
> I have two satellite sensors giving me SST values on two different
> grids:
> sensor1 is 4320 lon points by 2160 lat points
> sensor2 is 4096 lon points by 2048 lat points
>
> I need to interpolate the sensor1 data onto the sensor2 grid. To do
> this I have found via this newsgroup a useful routine written way back
> in 1994 by Dan Bergmann: interp_sphere.pro
>
> The interp_sphere.pro routine is called in the following way:
> IDL> grid = INTERP_SPHERE(lat,lon,data)
> where
> lat: The latitudes on the grid where interpolated
> values are desired (in degrees)
> lon: The longitudes on the grid where interpolated
> values are desired (in degrees)
> data: An array (3,ndata) where ndata is the number of
> data points, and can be any number larger than N.
> each row of data should contain a longitude, a
> latitude, and a value to be interpolated.
>
> Therefore in my case:
> lat => (lat_sensor2[valid_lat_s2])
> lon => (lon_sensor2[valid_lon_s2])
> data =>
> (lat_sensor1[valid_lat_s1],lon_sensor1[valid_lon_s1],data_se nsor1[valid])
>
> My problem lies in finding the correct indices for the 1d arrays. i.e
> the indices: 'valid_lat_s2' ; 'valid_lon_s2' ; 'valid_lat_s1' ;
> 'valid_lon_s1'
>
> The valid index for the data_sensor1 array is easy enough to find:
> valid = where(data_sensor1 NE land)
>
> If have a grid_sensor2 array which acts as a land/sea mask, how can I
> get the
> 'valid_lat_s2' and 'valid_lon_s2' indices ?
>

> And finally returning to my original post, how do I relate the 'valid'
> indices of the data_sensor1 array to the 'valid_lat_s1' and
> 'valid_lon_s1' of the lat_sensor1 and lon_sensor1 arrays ?
>
> I hope I have managed not to confuse you too much. I realize that I
> might be just complicating my life, but I would rather hope that I
> might be almost there...
>
> Again thanks very much for your help.
>
> Best Regards,
> Pepe
>
>

So you have something like (with presumably higher resolution!)

```
   lon0 = findgen(360)
   lat0 = findgen(181)-90
   nx   = 360

   ; Your dimensions are different... Let's make the number of lons, nx.

   data = fltarr(360,181)

   w = where(... data is valid ...)

   lon1 = a set of longitudes you want to interpolate to
   lat1 = a set of latitudes you want to interpolate to

   ox = n_elements(lon1)
   oy = n_elements(lat1)

   a = fltarr(oy)+1
   olon1 = lon1 # a ;  Creates a 2D array of longitudes for each output point
   a = fltarr(ox)+1
   olat1 = a # lat1 ;  Ditto for lats

   ; Don't know if interp_sphere will handle 2-D output coordinates.
   ; if not then...
       olon1 = reform(olon1, ox*oy)
       olat1 = reform(olat1, ox*oy)

   odata = interp_sphere(olon1, olat1, [lon0[w mod nx], lat0[w /nx], data[w]])

       odata = reform(odata, ox, oy) ; if needed to get back to a 2-d array
```

Done...

Regards,
Tom McGlynn