Subject: Re: INTERPOLATION TECHNIQUES HELP
Posted by dan on Tue, 19 Nov 1996 08:00:00 GMT
View Forum Message <> Reply to Message

In article <56acrg$vuk@news.esrin.esa.it>, Marcos Portabella Arnus - RS/EM <marcos> writes:
|> Hi,
|>
|> I need some help about the idl interpolation functions. I have tried all of
|> them and I could not get any satisfactory result. I supose this is due to my
|> lack of knowledge about them and this is the reason I am writing to this
|> newsgroup list. I am using IDL version 4.0.1 (vms alpha). My problem is the
|> following:
|>
|> I have three data vectors. The first two are X and Y position coordinates
|> (longitude and latitude) and the third one is the magnitude mesured at each
|> point (in my case, wind measurements). These points have not any order
|> (irregularly gridded points) . In order to make an interpolation of these
|> points in a regurlar grid (with a x and y spacing of half degree, for example)
|> I have tried all three idl functions that allow this type of interpolation. The
|> first one, the TRIGRID function, is very fast but the results are poor
|> (moreover, at the grid border, where it oftenly needs to extrapolate, the
|> results are incredibily wrong, even if I set the EXTRAPOLATE keyword). The
|> second one, the KRIG2D function, gives much better results but the major
|> problem is the execution time: the time increases exponentially with the number
|> of points. I do not know if changing the function parameters (A, CO, C1) may
|> reduce the execution time, but as far as I have tested with the idl help
|> examples I could no get any better result. The third and last one, the
|> MIN_CURVE_SURF function, has the same time restriction as the KRIG2D function.
|>
|> I would like to know if there is any other interpolating function for
|> irregularly gridded points that gives better time results; or, if using more
|> addequatly the KRIG2D or MIN_CURVE_SURF functions I can get better time
|> results as well.
|>
|> Thank you very much,
|>
|>      Marcos Portabella
|>

Here is a routine I wrote which maps scattered data (longitude,latitude,value)
onto a regular lat,lon grid on a sphere.

```
; $ID$

;+
; Name:
;      INTERP_SPHERE
;
```

```
; PURPOSE:
;       This function maps scattered data defined by
;       (longitude,latitude,value) onto a regular, but not
;       neccessarily evenly spaced, grid whose coordinates are
;       also defined by longitude and latitude. The procedure searches
;       for the N (default = 5) closest data points to each grid
;       point and then averages these N data points weighted by
;       distance^power from the grid point to the particular data point.
;       Default is power=-1 which weights the points inversely by
;       distance. All distances are along great circles on a sphere
;       (the shortest distance between two points along the
;       surface of a sphere).
;
; CATEGORY:
;       Interpolation?
;
; CALLING SEQUENCE:
;       grid = INTERP_SPHERE(lat,lon,data)
;
; INPUTS:
;
;       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.
;
; KEYWORD PARAMETERS:
;
;       N:       The number of closest data points to be used
;                for each grid point interpolation. Default = 5
;
;       power:   The exponent for the distance weighting function.
;                Default = -1 (weighting inversely by distance).
;                An input of power=-.5 would weight inversely by the
;                square root of the distance.
;
;       latwt:   The weighting for the interpolation in the meridional
;                (North-South) direction. For negative power,
;                latwt > 1 produces a weighting with less latitude
;                influence. Default = 1
;
;       mask:    Mask for calculating grid values
```

```
;
;
; OUTPUTS:
;
;       grid:    An array of interpolated data values. It has dimensions
;                (nlon,nlat) where nlon is the number of entries in the
;                input lon, and nlat is the number of entries in the input
;                lat.
;
; EXAMPLE:
;
; MODIFICATION HISTORY:
;
;       written by:    Dan Bergmann dbergmann@llnl.gov  11/10/94
;-


FUNCTION INTERP_SPHERE,lat,lon,data,n=n,power=power,latwt=latwt

nlat = (size(lat))(1)
nlon = (size(lon))(1)
grid = fltarr(nlon,nlat)

if (not(keyword_set(n))) then n = 5
if (not(keyword_set(power))) then power = -1
if (not(keyword_set(latwt))) then latwt = 1
if (not(keyword_set(mask)))  then begin
  mask = intarr(nlon,nlat)
  mask(*,*) = 1
endif

dtr = !pi / 180.

; convert lat and lon to radians

latr = dtr * lat
lonr = dtr * lon

; convert the lat and lon of the data to radians

dlatr = dtr * data(1,*)
dlonr = dtr * data(0,*)

; calculate the cartesian coordinates of the data points
; assuming a unit sphere.

xdata = cos(dlatr) * sin(dlonr)
ydata = cos(dlatr) * cos(dlonr)
```

```
zdata = sin(dlatr)

for x=0,nlon-1 do begin

  sinlonr = sin(lonr(x))
  coslonr = cos(lonr(x))

  for y=0,nlat-1 do begin

;   check to see if this grid should be calculated

    if (mask(x,y) ne 0) then begin

;     calculate the cartesian coordinates of this particular
;     grid point.

      xorig = cos(latr(y)) * sinlonr
      yorig = cos(latr(y)) * coslonr
      zorig = sin(latr(y))

;     calculate the length squared of the cords connecting this grid
;     point to all the data points and then sort the data points by
;     these values.

      corddistsq = (xorig-xdata)^2+(yorig-ydata)^2+((zorig-zdata)*latwt)^2

      sortdist = (sort(corddistsq))(0:n-1)

;     if a data point lies directly on top of this grid point, then
;     assign that value to the grid point.
;     Otherwise calculate the n great circle distances and do a weighted
;     average of the data values.

      if ((corddistsq(sortdist))(0) eq 0) then begin

        grid(x,y) = data(2,(sortdist)(0))

      endif else begin

        grcirdis = asin(sqrt(corddistsq(sortdist))/2.)

        grid(x,y) = (total(data(2,sortdist) * grcirdis^power)) / total(grcirdis^power)

      endelse

    endif

  endfor
```

```
endfor

return,grid

end
```

--
 *********************************************************** ***
**  Dan Bergmann                          dbergmann@llnl.gov  **
**  Atmospheric Science Division       fax   (510) 422-5844  **
**  Lawrence Livermore National Lab    human (510) 423-6765  **