Subject: Gridding irregular data points for subsequen contouring
Posted by axel on Tue, 03 Sep 1991 20:51:45 GMT
View Forum Message <> Reply to Message

Does anybody have a routine that grids irregular data points for
subsequent contouring. The PV-WAVE users library has a routine called
grid.pro which says it will do that using an external routine.
It seems to be looking for something by the name of gridder. Since
i don't have such a program grid.pro is not of much use. Can anybody
point me as to where to find this mysterious gridder program or some
thing that will do the job.

Thanks. Axel

--
* Axel Schweiger, University of Colorado, Boulder, CIRES**********
* Tel: 303-492-2963   *
* Email: axel@frostbite.colorado.edu   *
*   schweiger@colorado.bitnet ******************************

Subject: Re: Gridding irregular data points for subsequen contouring
Posted by wagener on Tue, 10 Sep 1991 17:07:54 GMT
View Forum Message <> Reply to Message

Here are two routines that you might want to use as a guideline to
create your own. As always NO GUARANTEE.

```
------------------------------cut here-----------------------------
function powgrid,x,y,z,xo,yo,range,pow,w
;+
; NAME:
; powgrid
; PURPOSE:
; Create a 2-d matrix from vector data
; CATEGORY:
; Interpolation
; CALLING SEQUENCE:
; powgrid,x,y,z,xo,yo,range,pow[,w]
; INPUTS:
; x,y,z = input vector data
; xo,yo = axes definition for output matrix
; range = radius of the area of influence around each grid
;    point, this should be large enough to cover all
;    the gaps in the input data
; pow = exponent of the weighting function (w=distance^-pow)
; OPTIONAL INPUT PARAMETERS:
; None
```

```
; KEYWORD PARAMETERS:
; None
; OUTPUTS:
; return,zo = matrix of dimension n_elements(xo),n_elements(yo)
; OPTIONAL OUTPUT PARAMETERS:
; w = sum of the weights (high values indicate reliable
;    data, low values mean nearest vector to the given
;    grid point was far away)
; COMMON BLOCKS:
; None
; SIDE EFFECTS:
; None
; RESTRICTIONS:
; This is a conservative approach relative to say a cubic spline
; technique, which means that smooth surfaces are not very well
; reproduced and peaks and values are somewhat truncated, but on
; the other hand this won't create artificially high peaks.
; PROCEDURE:
; Sum all vectors that fall within a circle of radius 'range'
; around each grid point, giving them weights that drop of with
; distance as distance^-pow.
; MODIFICATION HISTORY:
; 1991-7-29: created by Rick Wagener (BNL)
;-
dmin=1.e-30^(1./pow)
range2=range*range
nx=n_elements(xo)
ny=n_elements(yo)
zo=fltarr(nx,ny)*0.
w=zo
for l=0,ny-1 do begin
  dy=y-yo(l)
  for k=0,nx-1 do begin
    dx=x-xo(k)
    d=dx*dx+dy*dy
    testrange=range2>min(d)
    i=where(d le testrange,ni)
    d=d(i)
    wkl=(d>dmin)^(-0.5*pow)
    wo=total(wkl)
    zo(k,l)=total(z(i)*wkl)/wo
    w(k,l)=wo
  endfor
endfor
return,zo
end
------------------------------cut here-----------------------------
function spheregrid,x,y,z,xo,yo,range,pow,w,sigma
```

```
;+
; NAME:
; spheregrid
; PURPOSE:
; Create a 2-d matrix (longitude,latitude) from vector data
; CATEGORY:
; Interpolation
; CALLING SEQUENCE:
; spheregrid,x,y,z,xo,yo,range,pow[,w,sigma]
; INPUTS:
; x,y,z = input vector data
; xo,yo = axes definition for output matrix
; range = radius of the area of influence around each grid
;    point, this should be large enough to cover all
;    the gaps in the input data
; pow = exponent of the weighting function (w=distance^-pow)
; OPTIONAL INPUT PARAMETERS:
; None
; KEYWORD PARAMETERS:
; None
; OUTPUTS:
; return,zo = matrix of dimension n_elements(xo),n_elements(yo)
; OPTIONAL OUTPUT PARAMETERS:
; w = sum of the weights (high values indicate reliable
;    data, low values mean nearest vector to the given
;    grid point was far away)
; sigma = standard deviations
; COMMON BLOCKS:
; None
; SIDE EFFECTS:
; None
; RESTRICTIONS:
; This is a conservative approach relative to say a cubic spline
; technique, which means that smooth surfaces are not very well
; reproduced and peaks and values are somewhat truncated, but on
; the other hand this won't create artificially high peaks.
; PROCEDURE:
; Sum all vectors that fall within a circle of radius 'range'
; (in spherical trigonometric distance on a sphere)
; around each grid point (longitude, latitude), giving them
; weights that drop of with distance as distance^-pow.
; MODIFICATION HISTORY:
; 1991-7-29: created by Rick Wagener (BNL)
; spheregrid computes a regular longitude,latitude grid from
; irregularly spaced data.  The weight that each point contributes to
; a specific grid point within 'range' depends on the -pow power of
; the spherical trigonometric distance (0-180 deg)
;-
```

```
degrad=atan(1.)/45.
dx=min([abs(xo-shift(xo,1)),abs(yo-shift(yo,1))])
dmin=0.5*dx*degrad
range2=range*degrad/dmin
nx=n_elements(xo)
ny=n_elements(yo)
zo=fltarr(nx,ny)*0.
w=zo
sigma=zo
cox=x*degrad
coxo=xo*degrad
ccoy=cos((90.-y)*degrad)
scoy=sin((90.-y)*degrad)
ccoyo=cos((90.-yo)*degrad)
scoyo=sin((90.-yo)*degrad)
for k=0,nx-1 do begin
  scoydx=scoy*cos(cox-coxo(k))
  for l=0,ny-1 do begin
    d=acos(ccoy*ccoyo(l)+scoydx*scoyo(l))
; weight all points inside of dmin equally (weight=1), so that some of
; weights approximates the number of points included in each pixel.
    d=(d/dmin)>1.
    testrange=range2>min(d)
    i=where(d le testrange,ni)
    d=d(i)
    zi=z(i)
    wkl=d^(-pow)
    wo=total(wkl)
    szw=total(zi*wkl)
    zkl=szw/wo
    sigma(k,l)=total(zi*zi*wkl)-szw*zkl
    zo(k,l)=zkl
    w(k,l)=wo
  endfor
endfor
i=where(w ge 1.00001,ni)
if ni le 0 then $
  print,'WARNING in ''spheregrid'': none of the sigmas are meaningful' $
else sigma(i)=sqrt(sigma(i)/(w(i)-1.))
return,zo
--
Tschuess ...rick...
```

Richard Wagener |
Bldg. 426, rm. 43 | IP: wagener@bnl.gov