Subject: Re: Regrid / Interpolation Question
Posted by Sean[1] on Mon, 26 Mar 2012 18:32:00 GMT
View Forum Message <> Reply to Message

As a follow up on this, I wrote a program to do the re-grid/interpolation that does not involve loops (see example in previous post, or in program header).

I would really appreciate any feedback on this, especially if anyone thinks there is a faster way to do what I've done here. I have some huge arrays I'm working with, so making this is as fast as possible is important to me!

Thanks,
Sean

```
function regrid2D, vin, yin, yout

  ;Purpose:
  ;
  ; Re-grid a 2-D array (vin) to the output grid (yout) using the array yin.
  ; This program assumes that yin is the same size as vin, and that the values of yin are ordered
along each row
  ;
  ; The interpolating
  ;
  ;Inputs:
  ; vin - 2D array (nx,ny) of values to re-grid
  ; yin - 2D array (nx,ny) of ordered values corresponding to elements of vin. Values must be
ordered along the x-dimension
  ; yout - 1D array (nz) of output values used to re-grid vin
  ;
  ;Return value:
  ; vout - A 2D array (nz, ny) of values
  ;
  ;Example:
  ;
  ; Lets say I have 3 temperature vs. height profiles. Each profile has 6 points in the vertical, so the
arrays are (6,3).
  ;
  ; IDL> temp = [ [270, 224.3, 200., 190., 210, 230.], [284,231, 206.5,
208,200.,190.],[300,280,230,220.,185.,200.]]
  ; IDL> height=[ [0.5,1,2.3,2.7,3.2,4], [0.,1.3,3.4,3.6,3.8,5.3], [1.,1.2,2.7,3.6,4.4,6]]
  ;
  ; I want to interpolate to interpolate the temperature to 2 new heights:
  ;
  ; IDL> heightout = [1.5, 4]
  ;
  ; So I call regrid2d
```

```
  ;
  ; IDL> newtemp = regrid2d( temp, height, heightout)
  ;


  szv = SIZE(vin)
  IF szv[0] NE 2 THEN BEGIN
    print, 'Vin must be a 2D array!'
    return, -1
  ENDIF
  nx = szv[1]
  ny = szv[2]

  nz = n_elements(yout)

  miny=min(yin, max=maxy)
  yinsc = (double(yin)-miny)/(maxy-miny)                    ;scale yin to the range 0-1, and make
double

  minxvy = min(yinsc,dim=1,max=maxxvy)                      ;Store the min/max of each row, to be
used in preventing extrapolation (see below)

  yinsc = yinsc + REBIN( REFORM(DINDGEN(ny),1,ny), nx, ny)      ;Add the row number so that
each row is higher than the previous. E.g., the first row goes from 0-1, the next row goes from 1-2,
...

  youtsc = (double(yout)-miny)/(maxy-miny)                  ;Scale yout to the range 0-1, the same
way as yin
  youtsc = REBIN( reform(youtsc,nz,1), nz,ny)               ;Recast the scaled output grid to be 2D
(nz, ny)

  ;********** We need to prevent extrapolation of the values of yout are outside the bounds of a
given row of yin        ***************
  ;********** for each row of the youtsc array, set any values that are outside of the values in the
corresponding row of yinsc to NAN ***************
  bd=where( (youtsc LT rebin(transpose(minxvy),nz,ny)) OR (youtsc GT
rebin(transpose(maxxvy),nz,ny)) , bdct)

  youtsc = youtsc + REBIN( REFORM(DINDGEN(ny),1,ny), nz, ny)      ;Add the row number so
that each row is higher than the previous. E.g., the first row goes from 0-1, the next row goes from
1-2, ...

  ;Do the interpolation (this is a streamlined version of what interpol does)
  s = VALUE_LOCATE(yinsc, youtsc) > 0L < (n_elements(vin)-2)
  vout = (youtsc-yinsc[s])*(vin[s+1] - vin[s])/(yinsc[s+1] - yinsc[s]) + vin[s]

  ;Alternative way (slower, I think) of doing the interpolation
;  vin = REFORM( vin, nx*ny, /overwrite)                    ;make vin and yinsc 1D arrays for feeding
```

```
in to interpol (does interpol actually need them to be 1D?)
;  yinsc = REFORM( yinsc, nx*ny, /overwrite)
;  vout = interpol( vin, yinsc, youtsc )
;  vin = reform( vin, nx, ny, /overwrite)                    ;Return vin to its original state

  return, vout

end
```