Subject: Re: Regrid / Interpolation Question
Posted by Kenneth P. Bowman on Fri, 23 Mar 2012 14:59:46 GMT
View Forum Message <> Reply to Message

In article < 25522340.354.1332458318275.JavaMail.geo-discussion-forums@vb at19 >,
Sean <seand13@gmail.com> wrote:

> All,
>
> I have what seems to be a straightforward re-gridding/interpolation problem,
> but AFAIK there is no built-in vectorized way to do this that avoids loops.
>
> Here's my inputs --
> vin and yin are arrays of size (ni, nj), and the values of yin are ordered
> along rows
> (e.g., yin[i+1,*] > yin[i,*] for 0 =< i =< (ni-2))
> yout is an array of length nk
>
> The looped version of the interpolation is the following:
>
> yout = fltarr(nk,nj)
> for j = 0, nj-1 do yout[*,j] = interpol(yin[*,j], vin[*,j], yout)
>
> Is there an elegant and/or built-in way to do this without involving a loop?
>
> I've written a somewhat convoluted program to do this without a loop, but it
> involves some transforming and doesn't seem very elegant. I'm happy to upload
> if someone wants to see it.
>
> Sean

Use INTERPOLATE instead of INTERPOL, and compute a 2-D array of
coordinates to match yin that contains the row index of each point.

Ken Bowman

Subject: Re: Regrid / Interpolation Question
Posted by Sean[1] on Fri, 23 Mar 2012 18:04:31 GMT
View Forum Message <> Reply to Message

> Use INTERPOLATE instead of INTERPOL, and compute a 2-D array of
> coordinates to match yin that contains the row index of each point.
>
> Ken Bowman

I can see the general idea for doing this with interpolate --The code should look something like

yinterpolates = REBIN(transpose(lindgen(nj)), n_elements(yout), nj)
yout = interpolate(yin, xinterpolates, yinterpolates)

but I don't quite get how to calculate the x-interpolates. The problem is that the values in vin, while ordered in each row, are not evenly spaced -- Am I missing something simple here?

Subject: Re: Regrid / Interpolation Question
Posted by Kenneth P. Bowman on Fri, 23 Mar 2012 18:20:30 GMT
View Forum Message <> Reply to Message

In article < 16619448.872.1332525871438.JavaMail.geo-discussion-forums@pb jk8 >,
 Sean <seand13@gmail.com> wrote:

>> Use INTERPOLATE instead of INTERPOL, and compute a 2-D array of
>> coordinates to match yin that contains the row index of each point.
>>
>> Ken Bowman
>
> I can see the general idea for doing this with interpolate --The code should
> look something like
>
> yinterpolates = REBIN(transpose(lindgen(nj)), n_elements(yout), nj)
> yout = interpolate(yin, xinterpolates, yinterpolates)
>
> but I don't quite get how to calculate the x-interpolates. The problem is
> that the values in vin, while ordered in each row, are not evenly spaced --
> Am I missing something simple here?

INTERPOLATE uses the concept of 'fractional coordinates', which you can think of as floating-point indices into the array.

If your tabulated points are not evenly spaced, you need to first reverse interpolate the desired output coordinates onto the unevenly spaced grid to get the fractional coordinates. That is, think of your unevenly spaced x's as a function of array index.

Then use those fractional coordinates to interpolate the dependent variable to the output points.

Since you are doing the interpolation row-by-row, the y-coordinate should be trivial. That is, it should just be the row index itself.

Ken Bowman

Subject: Re: Regrid / Interpolation Question

> INTERPOLATE uses the concept of 'fractional coordinates', which you can think
> of as floating-point indices into the array.
>
> If your tabulated points are not evenly spaced, you need to first reverse
> interpolate the desired output coordinates onto the unevenly spaced grid to get
> the fractional coordinates. That is, think of your unevenly spaced x's as a
> function of array index.

I understand the concept of fractional coordinates, but I still don't understand how to reverse interpolate without either a) using interpol(), or b) using a loop. Perhaps a more concrete example would help with this discussion:

Lets say I have 3 temperature vs. height profiles. Each profile has 6 points in the vertical, so the arrays are (6,3).

```
temp = [ [270, 224.3, 200., 190., 210, 230.], [284,231, 206.5, 208,200.,190.,110],$
[300,280,230,220.,185.,200.]]
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]]
nx = 6
ny = 3
```

I want to interpolate to interpolate the temperature to 2 new heights:

```
heightout = [1.5, 4]
nout = 2
```

So my output array should be (nout=2,ny=3). One looped way, using both INTERPOL and INTERPOLATE, would be

```
xinterpolates = fltarr(nout, ny)

for j =0, ny-1  do xinterpolates[*,j] = interpol( indgen(6), height[*,j], heightout)

yinterpolates = lindgen(nout,ny) / nout

tempout = interpolate( temp, xinterpolates, yinterpolates)
```

As I said earlier, the yinterpolates part is trivial, but I don't see how to reverse interpolate to get the xinterpolates without using a loop + INTERPOL().

In this particular example, I also don't see why it wouldn't just be faster to do

```
tempout = fltarr(nout, ny)
for j=0, ny-1 do tempout[*,j] = interpol( temp[*,j], height[*,j], heightout)
```

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
```