Subject: Re: Regrid / Interpolation Question
Posted by Sean[1] on Fri, 23 Mar 2012 18:53:46 GMT
View Forum Message <> Reply to Message

> 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)
```