Subject: Re: GRIDDATA woes
Posted by Kenneth P. Bowman on Mon, 03 Mar 2008 14:24:14 GMT
View Forum Message <> Reply to Message

In article
<57808cc6-8454-45f1-a104-50e465ef294c@v3g2000hsc.googlegroups.com>,
 "ben.bighair" <ben.bighair@gmail.com> wrote:

> I have seen a number of messages on the newsgroup about interpolation
> from an irregular grid to a regular one. None appear to address the
> issues around gridding on a sphere. I don't think I can use anything
> as simple as INTERPOLATE since the input array is sampled at irregular
> intervals.
>
> So how is this kind of interpolation supposed to be done?

If your grid is rectangular and separable (in the sense that all the
longitudes in each "column" of data are the same and all of the
latitudes in each "row" of data are same), even if the coordinates
are not regularly spaced, then it is actually quite
easy to interpolate to any set of points (regular or irregular) using
INTERPOLATE. This should be much faster than triangulating.

This problem looks just like the one David Fanning was working
on recently, and here is an outline of the solution

> Assuming that your data is 2-D (x = longitude and y = latitude), create
> the grids that you want to interpolate to
> nx = 360
> ny = 181

> Compute the "interpolation coordinates" from the original grid

> yj = j + (y - y_original[j])/(y_original[j+1] - y_original[j])
> Since the input and output grids are the same in the x-direction, you

> xx = REBIN(x, nx, ny, /SAMPLE)
> yy = REBIN(REFORM(yi, 1, ny), nx, ny, /SAMPLE)
> Then interpolate
> new = INTERPOLATE(original, xx, yy)

By happy chance, the interpolation chapter from my book is the sample
that is posted online here

  http://idl.tamu.edu/Book.html

Subject: Re: GRIDDATA woes
Posted by wgallery on Mon, 03 Mar 2008 19:00:55 GMT
View Forum Message <> Reply to Message

On Mar 2, 9:57 pm, "ben.bighair" <ben.bigh...@gmail.com> wrote:
> Hi All,
>
> I have been having a problem similar to this one...http://tinyurl.com/2spe3v
>
> The solution to the problem in the above posting was to use GRID_INPUT
> to filter and reorder the data *before* calling QHULL and GRIDDATA.
> That doesn't seem to be the case this time as I faithfully perform
> these steps. However, the error message indicates that it is
> something similar is going on.
>
> The big picture is that I have an irregular grid (actually it is
> regular in longitude but irregular in latitude) that I want to
> interpolate onto a regular grid. I have assembled a mockup of the
> situation in this procedure...http://www.tidewater.net/~pemaquid/counterclockw ise_fail.pro
>
> The error message when the above is run is ...
>
> SeaDAS> z=counterclockwise_fail()
> % GRIDDATA: Triangle 5 not in counterclockwise order.
> % GRIDDATA: Triangle 6 not in counterclockwise order.
> % GRIDDATA: Triangle 7 not in counterclockwise order.
> % GRIDDATA: Triangle 17 not in counterclockwise order.
> % GRIDDATA: Triangle 30 not in counterclockwise order.
> % GRIDDATA: Triangle 31 not in counterclockwise order.
> % GRIDDATA: Triangle 34 not in counterclockwise order.
> % GRIDDATA: Triangle 40 not in counterclockwise order.
> % GRIDDATA: Triangle 42 not in counterclockwise order.
> % GRIDDATA: Triangle 49 not in counterclockwise order.
>
> I have tried changing the values in the code to double. That results
> in a similar set of errors but for a different set of triangles.
>
> % GRIDDATA: Triangle 4 not in counterclockwise order.
> % GRIDDATA: Triangle 5 not in counterclockwise order.
> % GRIDDATA: Triangle 6 not in counterclockwise order.
> % GRIDDATA: Triangle 16 not in counterclockwise order.
> % GRIDDATA: Triangle 33 not in counterclockwise order.
> % GRIDDATA: Triangle 35 not in counterclockwise order.
> % GRIDDATA: Triangle 36 not in counterclockwise order.
> % GRIDDATA: Triangle 39 not in counterclockwise order.

> % GRIDDATA: Triangle 42 not in counterclockwise order.
> % GRIDDATA: Triangle 45 not in counterclockwise order.
>
> Bah!
>
> I have seen a number of messages on the newsgroup about interpolation
> from an irregular grid to a regular one. None appear to address the
> issues around gridding on a sphere. I don't think I can use anything
> as simple as INTERPOLATE since the input array is sampled at irregular
> intervals.
>
> So how is this kind of interpolation supposed to be done?
>
> Thanks!
> Ben
>
> ** Structure !VERSION, 8 tags, length=76, data length=76:
> ARCH STRING 'ppc'
> OS STRING 'darwin'
> OS_FAMILY STRING 'unix'
> OS_NAME STRING 'Mac OS X'
> RELEASE STRING '6.3'
> BUILD_DATE STRING 'Mar 23 2006'
> MEMORY_BITS INT 32
> FILE_OFFSET_BITS
> INT 64

Ben,

1. There is an error in your routine to generate longitude.

;;lon = FINDGEN(nLon)/(nLon-1) * PS[0] + lonRange[0]
lon = fINDGEN(nLon) * PS[0] + lonRange[0]

Otherwise, lon has only two unique points

2. Try using the first form of grid_data: without /sphere:

;filter and reorder the data
;;GRID_INPUT, lon, lat, zValue, xyz, newZ, /SPHERE, /DEGREES, EPSILON
= PS[0]/2.0
GRID_INPUT, lon, lat, zValue, x1, y1, newZ, /DEGREES, EPSILON = PS[0]/
2.0

;build the triangulation
;;QHULL, xyz[0,*], xyz[1,*], tr, /DELAUNAY
QHULL, x1, y1, tr, /DELAUNAY

```
;interpolate
;;Z = GRIDDATA(xyz[0,*], xyz[1,*], newZ, /SPHERE,/DEGREES, $
Z = GRIDDATA(x1, y1, newZ, /SPHERE,/DEGREES, $
   METHOD = "NaturalNeighbor", MISSING = !VALUES.F_NAN, $
   XOUT = oLon, YOUT = oLat, TRIANGLES = tr )
```

You will no longer get the dreaded "Triangle 0 not in counterclockwise order" error.

Cheers,

Bill Gallery

Subject: Re: GRIDDATA woes
Posted by David Fanning on Tue, 04 Mar 2008 15:35:55 GMT
View Forum Message <> Reply to Message

Kenneth P. Bowman writes:

> By happy chance, the interpolation chapter from my book is the sample
> that is posted online here
>
> http://idl.tamu.edu/Book.html

I have to say, I made the mistake, initially, of thinking that book was too basic for a lot of people, but I took it with me on a recent trip to Europe and I found a LOT of information in it that I had not fully appreciated before. I've moved it over closer to Digital Image Processing by Gonzalez and Woods on my shelf, so it's more accessible to me from my chair. ;-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: GRIDDATA woes
Posted by David Fanning on Tue, 04 Mar 2008 16:03:31 GMT
View Forum Message <> Reply to Message

Ben Tupper writes:

> I have been having a problem similar to this one... http://tinyurl.com/2spe3v
>
>
> The solution to the problem in the above posting was to use GRID_INPUT
> to filter and reorder the data *before* calling QHULL and GRIDDATA.
> That doesn't seem to be the case this time as I faithfully perform
> these steps. However, the error message indicates that it is
> something similar is going on.

I've decided it's time to write an article on this topic,
since I have reason to understand it better myself. But,
do you find the IDL documentation of these three routines
particularly abstruse? In the QHULL documentation there
is no indication which of the arguments are input and which
are output, which set properties verses return information,
etc. In the words of one of my favorite expressions from
across the pond, it's a "pig's breakfast". :-(

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: GRIDDATA woes
Posted by ben.bighair on Tue, 04 Mar 2008 16:05:59 GMT
View Forum Message <> Reply to Message

On Mar 3, 9:24 am, "Kenneth P. Bowman" <k-bow...@null.edu> wrote:
> In article
> < 57808cc6-8454-45f1-a104-50e465ef2...@v3g2000hsc.googlegroups .com >,
>
> "ben.bighair" <ben.bigh...@gmail.com> wrote:
>> I have seen a number of messages on the newsgroup about interpolation
>> from an irregular grid to a regular one. None appear to address the
>> issues around gridding on a sphere. I don't think I can use anything
>> as simple as INTERPOLATE since the input array is sampled at irregular
>> intervals.
>
>> So how is this kind of interpolation supposed to be done?
>

> If your grid is rectangular and separable (in the sense that all the
> longitudes in each "column" of data are the same and all of the
> latitudes in each "row" of data are same), even if the coordinates
> are not regularly spaced, then it is actually quite
> easy to interpolate to any set of points (regular or irregular) using
> INTERPOLATE. This should be much faster than triangulating.
>
> This problem looks just like the one David Fanning was working
> on recently, and here is an outline of the solution
>
>> Assuming that your data is 2-D (x = longitude and y = latitude), create
>> the grids that you want to interpolate to
>> nx = 360
>> ny = 181
>> x = FINDGEN(nx)
>> y = -90.0 + FINDGEN(ny)
>> Compute the "interpolation coordinates" from the original grid
>> j = VALUE_LOCATE(y_original, y)
>> yj = j + (y - y_original[j])/(y_original[j+1] - y_original[j])
>> Since the input and output grids are the same in the x-direction, you
>> don't need to do anything with x. Expand x and yi into 2-D arrays
>> xx = REBIN(x, nx, ny, /SAMPLE)
>> yy = REBIN(REFORM(yi, 1, ny), nx, ny, /SAMPLE)
>> Then interpolate
>> new = INTERPOLATE(original, xx, yy)
>
> By happy chance, the interpolation chapter from my book is the sample
> that is posted online here
>
> http://idl.tamu.edu/Book.html
>
> Ken Bowman

Thanks Bill and Ken,

I had scoured the c.l.i-p archives but never used "regridding" keyword which, in hindsight, seems the perfect keyword. The perils of keyword searches...

The discussion on INTERPOLATE that you reference (see http://tinyurl.com/38mr7k) is the first time I have ever "gotten" INTERPOLATE. Thank you! The function has always felt so awkward because the units x and y are in dimensions - it always left me feeling a little disconnected from the physical meaning. I'll get over it.

For my purposes the INTERPOLATE method is probably just the ticket,

but I do have this lingering question about the fact that the input values are drawn from the surface of a sphere. What are the conditions under which I do need to worry about it? Is it the spacing between the input values? The extend over the sphere? Some combination?

I have read that gridding can be as much art as science, but I would love to have some general principles that tell me when I can rely on one more than the other.

I agree with David, Ken's book is an excellent resource. Mine is well thumbed. Come to think of it, all my IDL books are well thumbed.

Cheers and thanks again,
Ben

Subject: Re: GRIDDATA woes
Posted by Brian Larsen on Tue, 04 Mar 2008 16:23:04 GMT
View Forum Message <> Reply to Message

> The discussion on INTERPOLATE that you reference (seehttp://tinyurl.com/38mr7k)
> is the first time I have ever "gotten" INTERPOLATE. Thank you! The
> function has always felt so awkward because the units x and y are in
> dimensions - it always left me feeling a little disconnected from the
> physical meaning. I'll get over it.

Not that this helps with any understanding but this is another interpolate()/interpol() example. I found and copied an example somewhere to build this and it is a decent example.

http://tiny.cc/9XX9y
or longer url
http://people.bu.edu/balarsen/Home/IDL/Entries/2008/1/15_int erpol()_versus_interpolate().html

Cheers,

Brian

Brian Larsen
Boston University
Center for Space Physics

Subject: Re: GRIDDATA woes

Posted by David Fanning on Tue, 04 Mar 2008 17:05:13 GMT
View Forum Message <> Reply to Message

Bill Gallery writes:

> Ben,
>
> 1. There is an error in your routine to generate longitude.
>
> ;;lon = FINDGEN(nLon)/(nLon-1) * PS[0] + lonRange[0]
> lon = fINDGEN(nLon) * PS[0] + lonRange[0]
>
> Otherwise, lon has only two unique points
>
> 2. Try using the first form of grid_data: without /sphere:
>
> ;filter and reorder the data
> ;;GRID_INPUT, lon, lat, zValue, xyz, newZ, /SPHERE, /DEGREES, EPSILON
> = PS[0]/2.0
> GRID_INPUT, lon, lat, zValue, x1, y1, newZ, /DEGREES, EPSILON = PS[0]/
> 2.0
>
> ;build the triangulation
> ;;QHULL, xyz[0,*], xyz[1,*], tr, /DELAUNAY
> QHULL, x1, y1, tr, /DELAUNAY
>
> ;interpolate
> ;;Z = GRIDDATA(xyz[0,*], xyz[1,*], newZ, /SPHERE,/DEGREES, $
> Z = GRIDDATA(x1, y1, newZ, /SPHERE,/DEGREES, $
> METHOD = "NaturalNeighbor", MISSING = !VALUES.F_NAN, $
> XOUT = oLon, YOUT = oLat, TRIANGLES = tr)
>
> You will no longer get the dreaded "Triangle 0 not in counterclockwise
> order" error.

Well, actually, I *did* still get the error. And I continued to
get it until I removed the SPHERE keyword from GRIDDATA, too.
Has anyone, anywhere, gotten spherical triangles to work on anything?

I don't know what to say about this in my article, except
not to ever use the SPHERE keyword if you expect to use
these routines successfully. Any ideas?

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.

Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: GRIDDATA woes
Posted by David Fanning on Tue, 04 Mar 2008 17:14:46 GMT
View Forum Message <> Reply to Message

Ben writes:

> For my purposes the INTERPOLATE method is probably just the ticket,
> but I do have this lingering question about the fact that the input
> values are drawn from the surface of a sphere. What are the
> conditions under which I do need to worry about it? Is it the spacing
> between the input values? The extend over the sphere? Some
> combination?

Well, I'm thinking about this now, too, and I'm just
thinking out loud, but I wonder if this is only important
if your input coordinates have already been projected
onto some kind of map projection. In which case, maybe
you could project them back into UV coordinates, do
the gridding there (without the dread SPHERE keyword
set), and then reproject them back to lat/lon coordinates.

Just an idea. I'd like to hear from someone with more
experience with this.

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: GRIDDATA woes
Posted by wgallery on Tue, 04 Mar 2008 17:43:26 GMT
View Forum Message <> Reply to Message

On Mar 4, 12:05 pm, David Fanning <n...@dfanning.com> wrote:
> Bill Gallery writes:
>> Ben,
>
>> 1. There is an error in your routine to generate longitude.

>
>> ;;lon = FINDGEN(nLon)/(nLon-1) * PS[0] + lonRange[0]
>> lon = fINDGEN(nLon) * PS[0] + lonRange[0]
>
>> Otherwise, lon has only two unique points
>
>> 2. Try using the first form of grid_data: without /sphere:
>
>> ;filter and reorder the data
>> ;;GRID_INPUT, lon, lat, zValue, xyz, newZ, /SPHERE, /DEGREES, EPSILON
>> = PS[0]/2.0
>> GRID_INPUT, lon, lat, zValue, x1, y1, newZ, /DEGREES, EPSILON = PS[0]/
>> 2.0
>
>> ;build the triangulation
>> ;;QHULL, xyz[0,*], xyz[1,*], tr, /DELAUNAY
>> QHULL, x1, y1, tr, /DELAUNAY
>
>> ;interpolate
>> ;;Z = GRIDDATA(xyz[0,*], xyz[1,*], newZ, /SPHERE,/DEGREES, $
>> Z = GRIDDATA(x1, y1, newZ, /SPHERE,/DEGREES, $
>> METHOD = "NaturalNeighbor", MISSING = !VALUES.F_NAN, $
>> XOUT = oLon, YOUT = oLat, TRIANGLES = tr)
>
>> You will no longer get the dreaded "Triangle 0 not in counterclockwise
>> order" error.
>
> Well, actually, I *did* still get the error. And I continued to
> get it until I removed the SPHERE keyword from GRIDDATA, too.
> Has anyone, anywhere, gotten spherical triangles to work on anything?
>
> I don't know what to say about this in my article, except
> not to ever use the SPHERE keyword if you expect to use
> these routines successfully. Any ideas?
>
> Cheers,
>
> David
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming:http://www.dfanning.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

My mistake: I removed /sphere everywhere except in the posted article.

Subject: Re: GRIDDATA woes
Posted by David Fanning on Tue, 04 Mar 2008 17:54:50 GMT
View Forum Message <> Reply to Message

Bill Gallery writes:

> My mistake: I removed /sphere everywhere except in the posted article.

Yes, I assumed so. But did you have any justification for it?
I mean, besides the fact that it wouldn't work otherwise?
Even hand waving would be welcome at this point. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: GRIDDATA woes
Posted by David Fanning on Tue, 04 Mar 2008 19:49:54 GMT
View Forum Message <> Reply to Message

Kenneth P. Bowman writes:

> This problem looks just like the one David Fanning was working
> on recently, and here is an outline of the solution
>
>> Assuming that your data is 2-D (x = longitude and y = latitude), create
>> the grids that you want to interpolate to
>> nx = 360
>> ny = 181

>> Compute the "interpolation coordinates" from the original grid

>> yj = j + (y - y_original[j])/(y_original[j+1] - y_original[j])

This works OK, I think, if the values you wish to interpolate
to are completely contained within the bounds of the original
vectors. But, suppose the original array was 180x90 and
I want to interpolate from 360x180. Then, the beginning
and ending values in the vectors I want to interpolate to
are outside the bounds of the original vectors. When
I go to find the "interpolation coordinates", I encounter

divide by zero errors and get infinities in my vectors.

Do you have a way of handling this situation? I mention
this because in the perverse CCCMA climate model I am
using, the longitude vector is evenly spaced, *except*
for the two values at either end of the vector. (Don't
ask me, I have no idea.) My "regularly spaced" interpolation
vector blows up on me at either end.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: GRIDDATA woes
Posted by David Fanning on Tue, 04 Mar 2008 21:14:37 GMT
View Forum Message <> Reply to Message

Kenneth P. Bowman writes:

>> Compute the "interpolation coordinates" from the original grid

>> yj = j + (y - y_original[j])/(y_original[j+1] - y_original[j])

To tell you the truth, I can't get this to work at all. :-(

```
IDL> lat = [-87.5, 50, 25, 0, 30, 45, 64, 87.5]
IDL> y = Scale_Vector(findgen(7), -87.5, 87.5)
IDL> j = Value_Locate(lat, y)
IDL> yj = j + (y - lat[j])/(lat[j+1] - lat[j])
% Program caused arithmetic error: Floating illegal operand
IDL> print, yj
     0.000000  0.212121  0.424242  3.00000  3.97222  5.70175 -NaN
```

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: GRIDDATA woes
Posted by wgallery on Tue, 04 Mar 2008 22:17:35 GMT
View Forum Message <> Reply to Message

On Mar 4, 12:54 pm, David Fanning <n...@dfanning.com> wrote:
> Bill Gallery writes:
>> My mistake: I removed /sphere everywhere except in the posted article.
>
> Yes, I assumed so. But did you have any justification for it?
> I mean, besides the fact that it wouldn't work otherwise?
> Even hand waving would be welcome at this point. :-)
>
> Cheers,
>
> David
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming:http://www.dfanning.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

The /sphere method returns cartesian coordinates: x^2+y^2+z^2 = 1.0. I knew I wanted the results on a grid of latitude and longitude so the method without /sphere seemed the way to go. I never tried the /sphere option and the other method worked well enough for my application.

On the other hand, I don't understand the implications of the /sphere option: without it, you appear to be interpolating on a flat surface. What happens near the poles? What about crossing the meridian from 359 deg to 0 deg? I never investigated these question. Perhaps you can elucidate them in your white paper.

Cheers,
Bill

Subject: Re: GRIDDATA woes
Posted by pgrigis on Tue, 04 Mar 2008 22:39:31 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Kenneth P. Bowman writes:
>
>>> Compute the "interpolation coordinates" from the original grid

>>> yj = j + (y - y_original[j])/(y_original[j+1] - y_original[j])
>

> To tell you the truth, I can't get this to work at all. :-(
>
> IDL> lat = [-87.5, 50, 25, 0, 30, 45, 64, 87.5]
> IDL> y = Scale_Vector(findgen(7), -87.5, 87.5)
> IDL> j = Value_Locate(lat, y)
> IDL> yj = j + (y - lat[j])/(lat[j+1] - lat[j])
> % Program caused arithmetic error: Floating illegal operand
> IDL> print, yj
> 0.000000 0.212121 0.424242 3.00000 3.97222 5.70175 -NaN
Try:

IDL> print,interpol(findgen(8),lat,y)
 0.00000 0.212121 0.424242 3.00000 3.97222
5.70175 7.00000

Cheers,
Paolo

>
> Cheers,
>
> David
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: GRIDDATA woes
Posted by ben.bighair on Tue, 04 Mar 2008 22:46:46 GMT
View Forum Message <> Reply to Message

On Mar 4, 4:14 pm, David Fanning <n...@dfanning.com> wrote:
> Kenneth P. Bowman writes:
>>> Compute the "interpolation coordinates" from the original grid
>>> j = VALUE_LOCATE(y_original, y)
>>> yj = j + (y - y_original[j])/(y_original[j+1] - y_original[j])
>
> To tell you the truth, I can't get this to work at all. :-(
>
> IDL> lat = [-87.5, 50, 25, 0, 30, 45, 64, 87.5]
> IDL> y = Scale_Vector(findgen(7), -87.5, 87.5)
> IDL> j = Value_Locate(lat, y)
> IDL> yj = j + (y - lat[j])/(lat[j+1] - lat[j])
> % Program caused arithmetic error: Floating illegal operand
> IDL> print, yj
> 0.000000 0.212121 0.424242 3.00000 3.97222 5.70175 -NaN

>

Hi,

I think it comes a little clearer for me to leave VALUE_LOCATE out of it. Instead simply normalize the output coords into the input coordinate space. Something like this...

```
ilon = Scale_Vector(findgen(10), 10.0, 70.0)
ilat = [-87.5, 50, 25, 0, 30, 45, 64, 87.5]

olon = Scale_Vector(findgen(12), 0.0, 90.0)
olat = Scale_Vector(findgen(7), -87.5, 87.5)

ix = (olon - min(ilon)) / (max(ilon) - min(ilon))
iy = (olat - min(ilat)) / (max(ilat) - min(ilat))

idim = [n_elements(ilon), n_elements(ilat)]
iZ = HANNING(idim[0], idim[1])

oZ = INTERPOLATE(iZ, ix, iy, /GRID, MISSING = !VALUES.F_NAN)
```

It doesn't address the question about assuming a flat dataspce when the data are really in spherical coordinates. You might have a good solution to transform the coordinates to a projected space then interpolate, but I really don't know for sure. Taking the data as delivered, it is not projected at all - it just comes with two of the three parts needed to specify spherical coords plus a data value. The radius is whatever you want (in this case I was hoping to be on earth).

Ben

Subject: Re: GRIDDATA woes
Posted by Kenneth P. Bowman on Wed, 05 Mar 2008 02:55:19 GMT
View Forum Message <> Reply to Message

In article <MPG.223755c5d615d3ab98a2aa@news.frii.com>,
David Fanning <news@dfanning.com> wrote:

> Kenneth P. Bowman writes:
>
>> This problem looks just like the one David Fanning was working
>> on recently, and here is an outline of the solution
>>
>>> Assuming that your data is 2-D (x = longitude and y = latitude), create
>>> the grids that you want to interpolate to

>>> nx = 360
>>> ny = 181

>>> Compute the "interpolation coordinates" from the original grid

>>> yj = j + (y - y_original[j])/(y_original[j+1] - y_original[j])
>
> This works OK, I think, if the values you wish to interpolate
> to are completely contained within the bounds of the original
> vectors. But, suppose the original array was 180x90 and
> I want to interpolate from 360x180. Then, the beginning
> and ending values in the vectors I want to interpolate to
> are outside the bounds of the original vectors. When
> I go to find the "interpolation coordinates", I encounter
> divide by zero errors and get infinities in my vectors.
>
> Do you have a way of handling this situation? I mention
> this because in the perverse CCCMA climate model I am
> using, the longitude vector is evenly spaced, *except*
> for the two values at either end of the vector. (Don't
> ask me, I have no idea.) My "regularly spaced" interpolation
> vector blows up on me at either end.

VALUE_LOCATE finds the index of the point less than or equal to the search
value. You are trying to interpolate exactly to the last point. This
code correctly computes the index of that point to be 7, but there
is no point 8 to use for the interpolation. This can be solved
like this

```
IDL> lat = [-87.5, 50, 25, 0, 30, 45, 64, 87.5]
IDL> y = Scale_Vector(findgen(7), -87.5, 87.499)    <------
IDL> j = Value_Locate(lat, y)
IDL> yj = j + (y - lat[j])/(lat[j+1] - lat[j])
IDL> PRINT, yj
      0.00000     0.212120     0.424240     0.636360      3.97220
      5.70171      6.99996
```

Unfortunately, INTERPOLATE does not extrapolate when you are outside
the domain of the function.

Ken

Subject: Re: GRIDDATA woes
Posted by Kenneth P. Bowman on Wed, 05 Mar 2008 03:02:39 GMT
View Forum Message <> Reply to Message

> Thanks Bill and Ken,
>
> I had scoured the c.l.i-p archives but never used "regridding" keyword
> which, in hindsight, seems the perfect keyword. The perils of keyword
> searches...
>
> The discussion on INTERPOLATE that you reference (see http://tinyurl.com/38mr7k)
> is the first time I have ever "gotten" INTERPOLATE. Thank you! The
> function has always felt so awkward because the units x and y are in
> dimensions - it always left me feeling a little disconnected from the
> physical meaning. I'll get over it.
>
> For my purposes the INTERPOLATE method is probably just the ticket,
> but I do have this lingering question about the fact that the input
> values are drawn from the surface of a sphere. What are the
> conditions under which I do need to worry about it? Is it the spacing
> between the input values? The extend over the sphere? Some
> combination?

Remember, interpolation is an approximation. You make assumptions about the behavior of a function between known (tabulated) points. Bilinear interpolation is the crudest possible interpolation scheme.

Doing bilinear interpolation on a sphere is also an approximation, as the bilinear interpolator is strictly defined in Cartesian coordinates.

If I remember correctly, though, your grid is rather finely spaced. On that scale it is probably a very good approximation to assume that the world is flat. If you are worried about errors at that level, or you have a better idea of the actual shape of your function, then you could use a higher-order interpolator. That would depend on your purpose. It is a good idea to think about these things, but then I'm not doing geodesy. :-)

Ken

Subject: Re: GRIDDATA woes
Posted by Kenneth P. Bowman on Wed, 05 Mar 2008 03:04:16 GMT
View Forum Message <> Reply to Message

In article <MPG.22371a2a8f84a2ac98a2a5@news.frii.com>,
David Fanning <news@dfanning.com> wrote:

> Kenneth P. Bowman writes:
>
>> By happy chance, the interpolation chapter from my book is the sample
>> that is posted online here
>>

>> http://idl.tamu.edu/Book.html
>
> I have to say, I made the mistake, initially, of
> thinking that book was too basic for a lot of people, but
> I took it with me on a recent trip to Europe and I found
> a LOT of information in it that I had not fully appreciated
> before. I've moved it over closer to Digital Image Processing
> by Gonzalez and Woods on my shelf, so it's more accessible
> to me from my chair. ;-)

Sad to say, I refer to the book more often than I would like to,
but at least I have written a lot of things down where I can find
them. :-)

Cheers, Ken

Subject: Re: GRIDDATA woes
Posted by Kenneth Bowman on Wed, 05 Mar 2008 14:14:03 GMT
View Forum Message <> Reply to Message

In article <MPG.223755c5d615d3ab98a2aa@news.frii.com>,
 David Fanning <news@dfanning.com> wrote:

> Do you have a way of handling this situation? I mention
> this because in the perverse CCCMA climate model I am
> using, the longitude vector is evenly spaced, *except*
> for the two values at either end of the vector. (Don't
> ask me, I have no idea.) My "regularly spaced" interpolation
> vector blows up on me at either end.

I forgot to add that global spectral models, of which the CCCMA
is an example, use a Gaussian grid in the latitudinal direction.
This allows the use of Gaussian quadrature to compute the
Legendre transforms that are essential to the way a spectral
model works.

The Gaussian grid does not have points at the poles. Sometimes
it is desirable to interpolate to a regular grid that does have
points at the poles. In this case, I think the best approach is
to treat the pole points as special. They can be calculated by
averaging all of the points in the northernmost and southernmost
rows of the Gaussian grid. That is, to estimate the value at
90 deg N, average the last row of points, which might be at
89.5 deg N. This makes good physical sense, because the
northernmost row of points actually lie in a circle surrounding
the pole.

If you chose your regular grid wisely (;-)), then the row of points closest to, but not at, the poles, will lie equatorward of the highest latitude points of the Gaussian grid.

If you need to interpolate to a row of points poleward of the highest latitude Gaussian points, then use your interpolated (averaged) value at the poles to extend the Gaussian grid to +/- 90 deg and do bilinear interpolation as usual.

If that isn't clear, I can draw some pictures.

Cheers, Ken
