
Subject: Re: Interpolation

Posted by [ben.bighair](#) on Sat, 12 Apr 2008 22:26:48 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Apr 12, 4:24 pm, tarequea...@gmail.com wrote:

> On Apr 12, 11:57 am, "ben.bighair" <ben.bigh...@gmail.com> wrote:

>

>

>

>> On Apr 11, 11:05 pm, tarequea...@gmail.com wrote:

>

>>> Hello All (IDL Gods),

>

>>> I am back with yet another problem.

>>> I know...I know...its friday night. I apologize for that. But I am

>>> really stuck here for a while.

>

>>> The problem:

>

>>> In a certain part of my code I need to do interpolation. The data that

>>> I am dealing with are from a XY grid. I need to convert them to polar

>>> coordinate. So, what I do is following:

>

>>> a. I generate a radius vector r_vec and a theta array containing

>>> values from zero to 2π .

>>> b. Now use the simple polar-to-rectangular coordinate transform i.e. x

>>> $= r \cos(\theta)$ etc.

>>> c. Using these values I have a xy grid generated through a known r -

>>> theta values.

>>> d. Now think of superimposing the new xy grid(lets call it $x'y'$) on

>>> to the old xy grid which contains the real data.

>>> e. This is the part where I need interpolation. I do interpolations to

>>> get the $x'y'$ values from the xy points.

>

>>> So here's the question:

>

>>> ----- Is there any elegant way of doing this coordinate

>>> transformation ? (And in case you are thinking, "well you already have

>>> the xy data, so why not just convert to r -theta?", I have to say that

>>> the interpolation method actually gives me a way nicer dataset).

>

>>> My 2nd trouble is, and this is probably the biggest and dumbest

>>> problem for me.

>

>>> ----- i was playing around with several interpolation routines from

>>> IDL. My boss's suggestion was to use 'bilinear'.

>>> but I thought to give others' a shot too. Problem is, when I am

>>> done with interpolation, result is nothing like what I was

```

>>> expecting. A run down version of the code is shown below:
>
>>> =====start of code=====
>
>>> Nth= 10.
>>> dth= 1/Nth
>>> r_vec= findgen(Nth)/Nth
>>> theta_vec = findgen(Nth)/Nth * 2.*!Pi
>
>>> for i=0L,Nth - 1 do begin
>
>>> x[i]= r_vec[i]* cos(theta_vec)
>
>>> endfor
>
>>> for j=0L,Nth - 1 do begin
>
>>> y[j]= r_vec[j]* sin(theta_vec)
>
>>> endfor
>
>>> ;print,y
>
>>> ;plot,x,y
>
>>> -----
>>> Now I create the 'main' dataset on which I am going to use
>>> interpolation scheme.
>
>>> rr = findgen(20.)/30.
>>> tht = findgen(20.)/30. *2*!Pi
>
>>> m = fltarr(20,20)
>
>>> for j=0,19 do begin
>>>   for i=0,19 do begin
>
>>>     m[i,j] = rr[i]*cos(tht[j]) + 5.*rr[i]*sin(tht[j])
>
>>>     ;print,i
>>>   endfor
>>> endfor
>
>>> m_p=bilinear(m,x,y)
>
>>> End
>>> =====End of Code=====
>

```

```

>>> The problem is, as I mentioned above, when I plot m and the
>>> interpolated m_p, they do not look like similar at all.
>
>>> Any help will be greatly appreciated.
>
>>> Thanks in advance.
>
>>> ~tareque
>
>> Hi,
>
>> Coordinate transforms are very easy with CV_COORD().
>
>> I don't understand where you want to go with the interpolation. The
>> input coordinates to BILINEAR are rectangular and formed as indexed
>> based (as in subscript indices in X and Y.) As near as I can
>> reconstruct, X ranges from -0.500000 to 0.728115 and Y ranges
>> from -0.760845 to 0.285317. My reconstruction of your code is
>> below.
>
>> Cheers,
>> Ben
>
>> PRO tareque
>
>> Nth= 10L
>> dth= 1.0/Nth
>> r_vec= findgen(Nth)/Nth
>> theta_vec = findgen(Nth)/Nth * 2.*!Pi
>> x = fltarr(nth)
>> y = fltarr(nth)
>
>> for i=0L,Nth - 1 do begin
>
>> x[i]= r_vec[i]* cos(theta_vec[i])
>
>> endfor
>
>> for j=0L,Nth - 1 do begin
>
>> y[j]= r_vec[j]* sin(theta_vec[j])
>
>> endfor
>
>> ;print,y
>
>> ;plot,x,y
>

```

```

>> ;-----
>> ;Now I create the 'main' dataset on which I am going to use
>> ;interpolation scheme.
>
>> rr = findgen(20.)/30.
>> tht = findgen(20.)/30. *2*!Pi
>
>> m = fltarr(20,20)
>
>> for j=0,19 do begin
>>   for i=0,19 do begin
>
>>     m[i,j] = rr[i]*cos(tht[j]) + 5.*rr[i]*sin(tht[j])
>
>>   ;print,i
>>   endfor
>> endfor
>
>> m_p=bilinear(m,x,y)
>
>> STOP
>> End
>
> hi Ben,
>
> Thanks for getting back to me on this.
>
> I guess I could not clearly state my problem before. So, I am going to
> give it another try:
>
> The data I deal with comes from a ccd camera and this goes some
> tinkering and tweaking (to account for background noise and stuff).
> Then this 'processed' data is in need for interpolation. why? well the
> reason being this, that these processed data are actually in XY
> frame.
> We want them to be on a polar coordinate. The reason I cannot use
> 'cv_coord' is because in that case I will get the data in a r-theta
> plane which is
> directly correlated to the experimental XY grid. If we use
> interpolation, then we can be on a XY grid (yes, it is XY grid) which
> is carefully 'designed' according to
> our 'own' r-theta values. So in this way we have more leverage over
> data manipulation.
>
> Was it any clearer than before?
>
> And one more thing, I could not find much changes in your version
> except for an inclusion of 'STOP'. Did i miss something here??

```

>
> Thanks again for your time.
> Much appreciated !
>
> ~ Tareque

Hi Again,

The only thing I did other than STOP (which I forgot to remove) was redefine X and Y, which in your example were missing. I thought it would be helpful for others if the whole thing worked. I should have been clearer.

Speaking of clearer... I think I am more confused now. You might need the intervention from the IDL gods that you originally petitioned. Regardless of the parts I don't understand, the thing I do get is that BILINEAR is looking for you to provide interpolation coords that fit within the dimensions of the input array to be sampled.

Take a peek at the second example in the online help for BILINEAR... (slightly modified here...)

```
P = FINDGEN(3,3)
IX = [[0.5, 1.9], [1.1, 2.2]] ;Define the X subscripts.
JY = [[0.1, 0.9], [1.2, 1.8]] ;Define the Y subscripts.
Z = BILINEAR(P, IX, JY, MISSING = !VALUES.F_NAN) ;Interpolate.
PRINT, P
PRINT, Z
```

P prints as...

0.00000	1.00000	2.00000
3.00000	4.00000	5.00000
6.00000	7.00000	8.00000

Z prints as ...

0.800000	4.60000
4.70000	NaN

Note that IX and JY are mostly falling within the 0-2 range of indices for a 3x3 array. BILINEAR can extrapolate - just remove the MISSING keyword - but it can only extrapolate a little ways. In your example IX are sampled at

0.00000	0.0809017	0.0618034	-0.0927051	-0.323607
-0.500000	-0.485410	-0.216312	0.247214	0.728115

and JY are sampled at ...

0.00000
0.0587785

0.190211
0.285317
0.235114
-4.37114e-08
-0.352671
-0.665740
-0.760845
-0.529007

Which to my way of thinking your are interpolating all around either side of the [0,0] location. Is that what you intended? It is possible that in my redo of your example I corrupted the X and Y values you intended. In which case I would be not helping very much at all!

Cheers,
Ben
