
Subject: interpolate

Posted by [vshvetsk](#) on Sat, 28 May 1994 09:54:11 GMT

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

I have a following question that kind of left me in a dark (too late to think)

I have a matrix,let's say 3x3 and let's say I want to interpolate the middle cell value bu using other cells: that is M22 is result of interpolating all data around it: would I says:

result=Interpolate(([1,2,3],[1,2,3]),[2],[2])?

And what if I wanted just to do Linear Inerpolation by row only?

Could someone please help? Thank you

Victor Shvetsky

Subject: Re: INTERPOLATE

Posted by [Craig Markwardt](#) on Thu, 18 May 2000 07:00:00 GMT

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

Michael Cugley <mjcugley@medphys.dundee.ac.uk> writes:

> Maybe it's me, but this seems wrong...

This time I think it's you :-)

> IDL> bleeble = interpolate(bar, y, y, /grid)

If you interpolate the original array FOO you will get what you want.
You had me "foo"led for a little while there.

Craig

--

Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response

Subject: Re: INTERPOLATE

Posted by [Michael Cugley](#) on Fri, 19 May 2000 07:00:00 GMT

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

Michael Cugley wrote:

[[Much foolishness]]

Oh, I am a turnip. I worked this out on the bus back home :) Turns out I thought I needed the array of indices to interpolate to go from 0->1 (0 being the start of the array to be interpolated, 1 being the end); where what I rather obviously wanted was to go from 0->(size of old array -1). Doh!

--
Michael Cugley (mjcugley@medphys.dundee.ac.uk)

Subject: Re: INTERPOLATE
Posted by on Wed, 06 Jun 2012 09:34:57 GMT
[View Forum Message](#) <> [Reply to Message](#)

Den onsdagen den 6:e juni 2012 kl. 10:42:40 UTC+2 skrev Laura:

> Hi,
>
> I've got a temporal serie, temperatures, and I've got some 0's I would like to interpolate so that the serie is as complete as possible, with no blank spaces, e.g. no ceros. Is the Interpolate function the correct one to employ in this case?? I'm having problems understanding the location array, I thought I had only to look for the 0's in my serie, for example by
>
> index= where (data eq 0, count)
>
> and then apply the interpolation function, I know I'm missing something but I simply don't get it.
>
> The example that is on help contents:
>
> p = FINDGEN(4,4)
> PRINT, INTERPOLATE(p, [.5, 1.5, 2.5], [.5, 1.5, 2.5], /GRID)
>
> and prints the 3 by 3 array:
>
> 2.50000 3.50000 4.50000
> 6.50000 7.50000 8.50000
> 10.5000 11.5000 12.5000
>
> corresponding to the locations:
>
> (.5,.5), (1.5, .5), (2.5, .5),
> (.5,1.5), (1.5, 1.5), (2.5, 1.5),
> (.5,2.5), (1.5, 2.5), (2.5, 2.5)
>
> what I know is that

```
>
> p[0,0]= 0.0000 or p[0,3]=12
>
> but what I don't get is why
>
> p[0.5,0.5]=0.00000
>
> or
>
> print, p[0.9,0.9]=0.0000
> Does that mean that p[0,0],p[0.5,0.5], and p[0.9,0.9] are equivalent??? How do I construct the
location array?
>
> Thank you
>
> Laura
```

Yes, p[0,0], p[0.5,0.5], and p[0.9,0.9] are equivalent because floating point indexes are turned into integers. So you are accessing the same element in the p array.

As for your own example, `interpol` might be easier to use than `interpolate`. Try this:

```
data=[1.,2.,3.,4.,0.,6.,7.,0.,9.]
data2=data
indx0=where(data eq 0.)
indx1=where(data ne 0.)
data2[indx0]=interpol(data[indx1],indx1,indx0)
print,data
print,data2
```

It will print out:

```
1.00000  2.00000  3.00000  4.00000  0.00000  6.00000
7.00000  0.00000  9.00000
```

and

```
1.00000  2.00000  3.00000  4.00000  5.00000  6.00000
7.00000  8.00000  9.00000
```

which I believe is what you wanted?

Subject: Re: INTERPOLATE

Posted by _____ on Wed, 06 Jun 2012 10:21:09 GMT

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

Den onsdagen den 6:e juni 2012 kl. 11:34:57 UTC+2 skrev Mats Löfdahl:

> Den onsdagen den 6:e juni 2012 kl. 10:42:40 UTC+2 skrev Laura:

>> Hi,

>>

>> I've got a temporal serie, temperatures, and I've got some 0's I would like to interpolate so that the serie is as complete as posible, with no blank spaces, e.g. no ceros. Is the Interpolate function the correct one to employ in this case?? I'm having problems understanding the location array, I thought I had only to look for the 0's in my serie, for example by

>>

>> index= where (data eq 0, count)

>>

>> and then apply the interpolation function, I know I'm missing something but I simply don't get it.

>>

>> The example that is on help contents:

>>

>> p = FINDGEN(4,4)

>> PRINT, INTERPOLATE(p, [.5, 1.5, 2.5], [.5, 1.5, 2.5], /GRID)

>>

>> and prints the 3 by 3 array:

>>

>> 2.50000 3.50000 4.50000

>> 6.50000 7.50000 8.50000

>> 10.5000 11.5000 12.5000

>>

>> corresponding to the locations:

>>

>> (.5,.5), (1.5, .5), (2.5, .5),

>> (.5,1.5), (1.5, 1.5), (2.5, 1.5),

>> (.5,2.5), (1.5, 2.5), (2.5, 2.5)

>>

>> what I know is that

>>

>> p[0,0]= 0.0000 or p[0,3]=12

>>

>> but what I don't get is why

>>

>> p[0.5,0.5]=0.00000

>>

>> or

>>

>> print, p[0.9,0.9]=0.0000

>> Does that mean that p[0,0],p[0.5,0.5], and p[0.9,0.9] are equivalent??? How do I construct the location array?

>>

>> Thank you

>>

>> Laura

```

>
> Yes, p[0,0], p[0.5,0.5], and p[0.9,0.9] are equivalent because floating point indexes are turned
into integers. So you are accessing the same element in the p array.
>
> As for your own example, interpol might be easier to use than interpolate. Try this:
>
> data=[1.,2.,3.,4.,0.,6.,7.,0.,9.]
> data2=data
> indx0=where(data eq 0.)
> indx1=where(data ne 0.)
> data2[indx0]=interpol(data[indx1],indx1,indx0)
> print,data
> print,data2
>
>
> It will print out:
>
>   1.00000   2.00000   3.00000   4.00000   0.00000   6.00000
>   7.00000   0.00000   9.00000
>
> and
>
>   1.00000   2.00000   3.00000   4.00000   5.00000   6.00000
>   7.00000   8.00000   9.00000
>
> which I believe is what you wanted?

```

Sorry, you said you have a temporal sequence, so you have a time coordinate as well. Try this instead:

```

data=[1.,2.,3.,4.,0.,6.,7.,0.,9.]
t=[.4,.5,.6,.7,.8,.9,1.0,1.1,1.2]
data2=data
indx0=where(data eq 0.)
indx1=where(data ne 0.)
data2[indx0]=interpol(data[indx1],t[indx1],t[indx0])

plot,t,data,psym=4,yrange=[-.1,10]
oplot,t,data2,psym=1

```