Subject: Re: Non-monotonic Abscissa values for IDL function SPLINE_P
Posted by Jeremy Bailin on Wed, 03 Jun 2009 14:52:07 GMT
View Forum Message <> Reply to Message

On Jun 2, 12:40 pm, Xavier Ceamanos García <xavier.ceama...@gmail.com>
wrote:
> Hi folks!
>
> I have a question regarding the IDL SPLINE_P function. It works as
> follows...
>
> SPLINE_P, X, Y, Xr, Yr, [INTERVAL]
>
> Where 'X' is the original abscissa vector (let's say 0:1:249, in my
> example) and 'Y' the vector I'd like to interpolate (n_elements(Y)
> =250, of course).
>
> Then 'Xr' and 'Yr' are the outputs containing the abscissa values of
> the interpolated function and the interpolated vector, respectively.
>
> The keyword INTERVAL sets the desired interval between interpolants.
>
> The problem appears when checking the output Xr and Yr size. Normally,
> their size should be (250-1)*INTERVAL+1. However, that is not the
> case. For INTERVAL=100 I get an output size equal to 25147 instead of
> 24901. The reason is that the output abscissa is not monotonic and
> that does allow me to continue working with the Y output.
>
> Does anyone know how to make it monotonic?
>
> I would like to use this function instead of SPLINE since the latter
> is bloody slower than the the first!
>
> I thank you all in advance,
>
> Cheers,
>
> Xavi

Two things here...

First, I think that INTERVAL is in physical units, so the number of
elements you get out for a given value of INTERVAL depends on what Y
does (i.e. if you double the actual values of Y, the number of output
points should approximately double). The actual number can vary a bit
from a simple calculation because SPLINE_P seems to always put output
values at each input value, and then within each interval it puts a
fixed number of output points between the input points that are spaced

equidistantly.

Second, if Xr ends up being non-monotonic, it's because that's what the spline does... you can't force it to be monotonic if that's not the solution. Why do you need it to be monotonic? (for interpolation, maybe?) If you want to excise any times when Xr goes backwards, you could do something like this:

```
nxr = n_elements(Xr)
goodpoints = where(Xr ge max(total(identity(nxr),/cumul,2) * rebin
(Xr,nxr,nxr), dimen=1))
```

...and then just use Xr[goodpoints] and Yr[goodpoints]. But that may not really be what you want - if X is monotonic but Xr isn't, that may well be a sign that your spline is not really a good interpolating function. For example, I was playing around with the following:

```
X = findgen(5)
Y = [0., 100., 100., 0., -100.]
SPLINE_P, X, Y, Xr, Yr, INTERVAL=10
PLOT, X, Y, XRANGE=[-20,20]
OPLOT, Xr, Yr, PSYM=-4, LINES=2
```

You can see that the spline is pretty disastrous in this case (and the "excise all points that aren't greater than the cumulative maximum" approach that I listed above won't give you anything remotely like the data points).

-Jeremy.

Subject: Re: Non-monotonic Abscissa values for IDL function SPLINE_P
Posted by Xavier Ceamanos Garci on Wed, 03 Jun 2009 15:50:10 GMT
View Forum Message <> Reply to Message

Thank you so much Jeremy for the answer!

You are very right. I use splines to interpolate spectra. Then, the main goal is to re-sample these spectra. Each value of a spectrum corresponds to a given wavelength and in my case all points are separated by a constant wavelength distance. Hence, it is crucial to know which wavelengths corresponds to each point of the over-sampled spectra. A good re-sampling is not possible otherwise.

Then, are you telling me that the number of points of the output vector depends on the input data? That would mean that it would be different for each spectrum interpolation...

So far, I am using the "SPLINE" function which produces a monotonic output. In this case, it is easy to know the output wavelengths. It is slower though...

I was just wandering if there is any way to get the same results I get with SPLINE but using SPLINE_P...

Thanks again!

Xavi

Subject: Re: Non-monotonic Abscissa values for IDL function SPLINE_P
Posted by Jeremy Bailin on Wed, 03 Jun 2009 16:00:26 GMT
View Forum Message <> Reply to Message

> nxr = n_elements(Xr)
> goodpoints = where(Xr ge max(total(identity(nxr),/cumul,2) * rebin
> (Xr,nxr,nxr), dimen=1))

Incidentally, if nxr is large, that code will run out of memory pretty quickly and you're better off with an iterative solution like this:

```
goodpoints = lindgen(n_elements(Xr))
repeat begin
  nowinc = where(Xr[goodpoints[1:*]] gt Xr[goodpoints], ninc,
ncomp=ndec)
  if ninc gt 0 then goodpoints = [0,goodpoints[nowinc+1]]
endrep until ndec eq 0
```

-Jeremy.

Subject: Re: Non-monotonic Abscissa values for IDL function SPLINE_P
Posted by Jeremy Bailin on Wed, 03 Jun 2009 18:31:01 GMT
View Forum Message <> Reply to Message

On Jun 3, 11:50 am, Xavier Ceamanos García <xavier.ceama...@gmail.com> wrote:
> Thank you so much Jeremy for the answer!
>
> You are very right. I use splines to interpolate spectra. Then, the
> main goal is to re-sample these spectra. Each value of a spectrum
> corresponds to a given wavelength and in my case all points are
> separated by a constant wavelength distance. Hence, it is crucial to
> know which wavelengths corresponds to each point of the over-sampled
> spectra. A good re-sampling is not possible otherwise.

>
> Then, are you telling me that the number of points of the output
> vector depends on the input data? That would mean that it would be
> different for each spectrum interpolation...
>
> So far, I am using the "SPLINE" function which produces a monotonic
> output. In this case, it is easy to know the output wavelengths. It is
> slower though...
>
> I was just wandering if there is any way to get the same results I get
> with SPLINE but using SPLINE_P...
>
> Thanks again!
>
> Xavi

Xavi,

Yes, the number of points in the output vector depends on the input data. For example:

```
IDL> x = findgen(20)
IDL> y = sin(0.5*x)
IDL> spline_p, x, y, xr, yr, interval=0.25
IDL> help, xr, yr
XR              FLOAT     = Array[96]
YR              FLOAT     = Array[96]
IDL> y = sin(0.5*x)+x
IDL> spline_p, x, y, xr, yr, interval=0.25
IDL> help, xr, yr
XR              FLOAT     = Array[120]
YR              FLOAT     = Array[120]
```

So it will indeed be different for each spectrum. But your code should be general enough to deal with that using n_elements(xr).

You might want to directly use SPL_INIT and SPL_INTERP... I suspect that may do what you want.

-Jeremy.