Subject: polar interpolation
Posted by Thomas Gutzler on Fri, 10 Jan 2003 03:20:15 GMT
View Forum Message <> Reply to Message

Good morning,

I am looking for a function that can do a polar interpolation of a
[2,n]-array.
What I don't want is to convert polar koordinates to rect, interpolate,
and reconvert them to polar.

I'm not sure if the given IDL-functions can do this and unfortunatiely I
couldn't google anything that helps. Neither davids nor RSIs page could
help :(

Tom

Subject: Re: polar interpolation
Posted by Thomas Gutzler on Mon, 13 Jan 2003 06:43:24 GMT
View Forum Message <> Reply to Message

Thomas Gutzler wrote:
>
> I should write both functions, compare, and then decide again if I want
> to use the conversion-method. Just wanted to know _if_ there is another
> way to do it.

I'm facing the problem now. I can see it really clear and it won't let
me pass.
I want an interpolated curve in polar coordinates AND equidistant
theta-values. Since the original curve isn't a straight line it's really
complicated to pass the correct x-values to interpol so that
reconversion of the interpolated curve would have eqidistant
theta-values (of type integer).
Can anybody see and solve the problem or even give me a hint?

Tom

Subject: Re: polar interpolation
Posted by James Kuyper on Mon, 13 Jan 2003 17:40:05 GMT
View Forum Message <> Reply to Message

Stein Vidar Hagfors Haugan wrote:
>
> James Kuyper <kuyper@saicmodis.com> writes:

>
>> Thomas Gutzler wrote:
>>>
>>> Good morning,
>>>
>>> I am looking for a function that can do a polar interpolation of a
>>> [2,n]-array.
>>> What I don't want is to convert polar koordinates to rect, interpolate,
>>> and reconvert them to polar.
>>
>> If you have data that comes close to the pole, that's precisely what you
>> should do. Otherwise, you're going to see some very bizarre results in
>> that vicinity. The pole is a singular point in that coordinate system,
>> and you can only approach it by using a coordinate system where it isn't
>> a singular point.
>>
>> If you don't come close to the pole, you should be able to use ordinary
>> interpolation routines, treating rho, theta as if they were x and y.
>> That won't produce exactly the right results, but anything that produces
>> exactly the right results is going to be mathematically equivalent to
>> converting back to rectangular coordinates.
>
> Wouldn't it be better to do the interpolation close to the pole in a
> rotated (i.e. translated) polar coordinate system? Tilt the polar axis
> by 90 degrees, interpolate, tilt back?

That would work, but it has no advantages over converting to rectangular coordinates, and it has the disadvantage of treating near-polar data differenently from other data. The conversion from polar coordinates with one pole to polar coordinates with a different pole is no simpler than the conversion to rectangular coordinates. In fact, the easiest way to do the conversion is to use rectangular coordinates as an intermediate step. Rotation around a polar axis is simple; moving the polar axis is not.

--
James Kuyper
MODIS Level 1 Lead
Science Data Support Team
(301) 352-2150

Subject: Re: polar interpolation
Posted by Rick Towler on Mon, 13 Jan 2003 18:39:29 GMT
View Forum Message <> Reply to Message

"Thomas Gutzler" <tgutzler@ee.uwa.edu.au> wrote

> Thomas Gutzler wrote:
>>
>> I should write both functions, compare, and then decide again if I want
>> to use the conversion-method. Just wanted to know _if_ there is another
>> way to do it.
>
> I'm facing the problem now. I can see it really clear and it won't let
> me pass.
> I want an interpolated curve in polar coordinates AND equidistant
> theta-values. Since the original curve isn't a straight line it's really
> complicated to pass the correct x-values to interpol so that
> reconversion of the interpolated curve would have eqidistant
> theta-values (of type integer).
> Can anybody see and solve the problem or even give me a hint?

I might be missing something here (I usually am) but why won't simple linear interpolation work? If it is difficult to get one of the canned routines to work, brew your own:

Interpolate between points a and b:

iFactor = (Ti - Ta) / (Tb - Ta)

Ri = Ra + (iFactor * (Rb - Ra))

Where Ti is the Theta value where you are interpolating your radius Ri.

-Rick

Subject: Re: polar interpolation
Posted by Thomas Gutzler on Tue, 14 Jan 2003 01:36:16 GMT
View Forum Message <> Reply to Message

Hi,

Rick Towler wrote:
> "Thomas Gutzler" <tgutzler@ee.uwa.edu.au> wrote
>>
>> I want an interpolated curve in polar coordinates AND equidistant
>> theta-values. Since the original curve isn't a straight line it's really
>> complicated to pass the correct x-values to interpol so that
>> reconversion of the interpolated curve would have eqidistant
>> theta-values (of type integer).
>
> I might be missing something here (I usually am) but why won't simple linear

> interpolation work?

Hm, because I want it to be a spline or sth.

> If it is difficult to get one of the canned routines to
> work, brew your own:
>
> Interpolate between points a and b:
> iFactor = (Ti - Ta) / (Tb - Ta)
> Ri = Ra + (iFactor * (Rb - Ra))
> Where Ti is the Theta value where you are interpolating your radius Ri.

Thanx for this, but..
what I'm trying to do is to create the fitting function using spline and
solve this for the desired theta-values.
If I can't get this to work I will perhaps use your linear one.

Tom

Subject: Re: polar interpolation
Posted by Thomas Gutzler on Mon, 20 Jan 2003 03:33:54 GMT
View Forum Message <> Reply to Message

Hi,

Rick Towler wrote:
>
> Interpolate between points a and b:
>
> iFactor = (Ti - Ta) / (Tb - Ta)
> Ri = Ra + (iFactor * (Rb - Ra))
> Where Ti is the Theta value where you are interpolating your radius Ri.

I stole this from interpol.pro. The function is the same, just other
names for variables:

```
;    ,------------------------------------------------------------ -----
;   | Purpose:
;   | linear, polar interpolation for irregular grids
;   | T: The theta values (in degrees) for R (radius). This vector
;   | must have same # of elements as R. The values MUST be
;   | "monotonically ascending" or "descending" (see problem below).
;   | U: The theta values for the result. The result will have
;   | the same number of elements as U. U does not need to be
;   | monotonic. If U is outside the range of T, then the
;   | closest two endpoints of (T,R) are linearly extrapolated.
;    `------------------------------------------------------------ ------
FUNCTION polar_interpol, R, T, U
```

```
  m = N_elements(R)                  ; # of input pnts
  s = VALUE_LOCATE(T, U) > 0L < (m-2)  ; Subscript intervals.
  p = ( U - T[s] ) * ( R[s+1] - R[s] ) / ( T[s+1] - T[s] ) + R[s]
  RETURN, p
END
```

But what to do, if an array like this wants to be interpolated:
R: [1,2,3]
T: [340,350,20]
U: [355,5,15,25]

I can see 2 solutions at the moment:

1) - find the first index (i) after crossing the magic 0-line.
     In this case: i_T = 2 and i_U = 1
   - T[i_T:*] = T[i_T:*] + 360
   - U[i_U:*] = U[i_U:*] + 360
   - interpolate T: [340,350,380], U: [355,365,375,385]
   - RESULT = [2.16667, 2.50000, 2.83333, 3.16667]

2) - split the vectors
     R1: [1,2,2.33333] (last value must be interpolated)
     T1: [340,350,360]
     U1: [355]
     R2: [2.33333,3]
     T2: [0,20]
     U2: [5,15,25]
   - interpolate separately
     RESULT1: [2.16667]
     RESULT2: [2.50000, 2.83333, 3.16667]
   - return [RESULT1, RESULT2]

Is there a trick to do this easier, using some strange functions
(VALUE_LOCATE was strange for me) ?
Keep in mind that the next step would be a quadratic interpolation.

thanks, Tom