Subject: Re: weird behavior of Triangulate
Posted by Yngvar Larsen on Wed, 12 Sep 2012 10:12:39 GMT
View Forum Message <> Reply to Message

On Tuesday, 11 September 2012 21:53:41 UTC+2, David Fanning wrote:
> Yngvar Larsen writes:
>
>> ;; Calculate output map grid values in input coordinates
>> latlon = map_proj_inverse(mapx, mapy, map_structure=map)
>
> I don't understand this step. In your explanation of the
> "right way" you say:
>
> "Start with a regular grid in the _output_ domain.
> Calculate where these grid points are located in
> the input domain."
>
> I would have thought this means "use the map structure
> of the input data" in this case.

I see. This example is a bit misleading, since the input grid is lat/lon. In general, it would look something like this:

```
;; Example: conversion from one UTM zone 33 to 34
map1 = map_proj_init('UTM', zone=33, datum=8)
map2 = map_proj_init('UTM', zone=34, datum=8)

;; Define output grid (UTM Z34 coordinates)
mapx = ...
mapy = ...

;; Calculate input grid coordinates for output grid
latlon = map_proj_inverse(mapx, mapy, map_structure=map2)
utm33 = map_proj_forward(latlon[1,*], latlon[0,*], map_structure=map1)

;; xcoord_in/ycoord_in are the coordinates of the grid points of your input grid.
;; dx_in/dy_in is the grid resolution in x/y direction.
xind = (utm33[0,*] - xcoord_in[0,0])/dx_in
yind = (utm34[1,*] - ycoord_in[0,0])/dy_in
;; or
;;   yind = (ycoord_in[0,0] - coord[1,*])/dy_in
;; if first line of the input data array is "upper row"/"northernmost row" like in your example
```

> And, yet, you are using the map structure of the output data grid.

Yes, that is the whole point! However, as I showed above, if the input grid is not lat/lon, you need also the map structure for the input data grid.

> Yes, the points will all fall within the input domain, since
> that is a global domain. But, what about when the
> input domain is NOT global, so that some of the points
> are inside the output domain, but some are outside, too?

I'm not sure if my brain parsed that sentence correctly, but I think I know what you mean. I omitted some details in my program:

* how to handle points in your defined output grid that fall outside the available input grid

In your example, the input grid was global, so no problem. For the nearest neighbour interpolation, you have to check if the calculated XIND and YIND are inside the interval [0, X/YSIZE-1] and handle the points that are not separately, e.g. insert NaN or something else in your output grid. For the other two, use the MISSING keyword to INTERPOLATE.

* how to handle missing data in your input grid (NaNs, "magic" values like -9999, etc). I.e. _almost_ regular input grid.

Your example did not have this problem. Interpolation might be tricky in this case. This is left as an exercise for the reader :)

* If you don't want to define the output grid explicitly, how to calculate automatically an output grid that covers the available input data.

For your example, where the input data are global, I'm not sure if this is even possible. But for most cases, you just use map_proj_forward(..., map=inputmap)+ map_proj_inverse(..., map=outputmap), and use MAX/MIN + FLOOR/CEIL on the resulting coordinates.

--
Yngvar