Subject: Re: weird behavior of Triangulate
Posted by DavidF[1] on Mon, 10 Sep 2012 15:26:29 GMT
View Forum Message <> Reply to Message

Yngvar Larsen writes:

> So your problem is basically that GRIDDATA is slow?

"Ungodly slow" is what I think I said. ;-)

> From my point of view, GRIDDATA is for gridding _irregular_ data, which is a hard problem. If your data is already on some regular grid, why would you want to triangulate? Regular interpolation is all that is needed if you do it the right way.

It would be wonderful to give people a pointer to the "right way"
then. I've apparently been doing it the "wrong way" because I
can't get it to work properly, even though I have been working on
the problem, off and on, for years.

> In general, I do the following to transform data from one grid to another:
>
>
>
> 1) Divide the _output_ grid in manageable blocks. What "manageable" means, will depend on many things, but for satellite data mostly on memory limitations. Overlap between the blocks might be necessary if you are going to include filtering/interpolation.

It would be useful to have a robust algorithm for doing this kind of
chunking. Or, in the absence of that, at least some general rules of
thumb that people could use. I've never seen anything written down
about this.

> Repeat the following for your each of the output blocks:
>
> 2a) Calculate coordinates in _input_ grid corresponding to the grid points in your current output grid.
>
> 2b) Convert these coordinates to indices in the input data grid, including subpixels.
>
> 2c) Extract from your input dataset a "big enough" tile from the input grid. Or keep the entire input dataset in memory if it is already small enough.
>
> 2d) Choose gridding mode. If your output grid resolution is approximately the same or higher than the input grid resolution, nothing is needed here. However, if the output resolution is coarser, you might want to do something to avoid undersampling. E.g., smooth input data to match output resolution. Of course, if you have some missing data in the input grid, you must be careful at this point.
>

> 2e) Perform the actual regridding, using mapping indices from (2b), say XIND and YIND.
>
> Nearest neighbour: outdata = indata[round(xind), round(yind)]
>
> Bilinear interpolation: outdata = interpolate(indata, xind, yind)
>
> Cubic interpolation: outdata = interpolate(indata, xind, yind, cubic=-0.5)
>
> Again, take care if you have missing data in your input.

> 3) Glue together output blocks after elimination of possible block overlap.

I appreciate the help. I'll see if I can find some time to have another
go at this. I have been working on a "map_patch" alternative that
sorta works. Perhaps I can fit these ideas into it in a reasonably
robust fashion.

Cheers,

David