Subject: Re: spatial interpolation
Posted by Mark Hadfield on Tue, 30 Sep 2003 00:40:49 GMT
View Forum Message <> Reply to Message

Isa Usman wrote:
> Sorry, I should have really said that the data I am interpolating from
> (radar data) is on a polar grid. But because the data does not have a
> central node due to only data within a radial distance of 20km and 40km
> being available, I termed it as irregularly gridded. The data spans out in
> ~0.25 degree increments up to an angle of 50 degrees. The points I am
> interpolating to have a central node situated on the centre of the plane
> defined by the radar data points.
>
> What I did in the program was to histogram the original points and the
> points that I wanted to interpolate to over a certain rectangular area.
> Essentially this constructed a mesh grid over the points and then it would
> loop over each grid to do the interpolation. I did this so that i wouldn't
> need to loop over every point to interpolate. To make sure there weren't any
> "edge effects" in the interpolation, either 8 or 24 grids surrounding the
> main grid were joined together before interpolating using MIN_CURVE_SURF.

Hmm, I think I understand that. But I'm afraid understanding how other
people do things is *really* not my strong point, so I'll just set down
some thoughts and principles that might be helpful.

- The best IDL procedure for interpolating from irregularly gridded data
is GRIDDATA, introduced in 5.5. Notwithstanding what it says in the
first paragraph of the documentation, it can interpolate to any output
grid, irregular or otherwise.

- When you have good data coverage and you want to interpolate within
the area bounded by the data points, the most robust method, if not the
most accurate, is probably linear interpolation.

- Linear interpolation with GRIDDATA requires TRIANGLES data to be
supplied (as do several other methods). The TRIANGLES data defines how
to connect the points before interpolating. GRIDDATA then determines
which triangle each output point is contained in and interpolates
linearly to it from the three vertices.

- For scattered input data you can generate a triangulation with the
TRIANGULATE routine. This can be quite time consuming. If your input
data are on a regular grid then you can construct a triangulation to
suit the grid. For example, I have a routine that constructs a
triangulation for a rectangular mesh.

- GRIDDATA can also carry out minimum curvature surface interpolation,
as does the older MIN_CURVE_SURF fuction. This method is very expensive

on large data sets: execution time increases with the cube of the number
of input data points. You can avoid this expense by subdividing your
domain (as you seem to be doing), but GRIDDATA also allows you to cut
execution time drastically by specifying the MIN_POINTS keyword.

- In my line of business (ocean hydrodynamic modelling) I interpolate a
lot between rectilinear and curvilinear grids (the latter being a mesh
of rectangular cells like the former, but rotated or distorted). I like
to do this as a two-stage process: 1) locate each output data point in
the "index space" of the input grid; 2) do the interpolation with
INTERPOLATE. There are a couple of reasons for doing it this way: 1) it
is cheaper than using GRIDDATA if I want to carry out several
interpolations on the same pair of grids; 2) missing data can be handled
better. My Motley library at
http://www.dfanning.com/hadfield/README.html has examples of this approach.

Good luck!

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)