Subject: TRIANGULATE. Finding contiguous cells efficiently?
Posted by Libertan on Sat, 24 Feb 2007 02:39:39 GMT
View Forum Message <> Reply to Message

Re: finding a cell's three contiguous neighbours in a Delaunay
Triangulation.

Dear IDL users,

I have searched this group, but cannot seem to find the solution to my
problem.

TRIANGULATE is a wonderful function, and is very fast indeed. I would
like to write a comparably expeditious routine for finding each
triangular cell's three contiguous (edge-sharing neighbour) cells. I
would imagine that this is done frequently, and is conceptually rather
staright forwards.

TRIANGULATE,x,y,TR

I understand that TR is a [3,N] array which essentially tells me the
three vertices of each of the N triangular Delaunay cells. It
certainly contains sufficient information to find an interior cell's 3
contiguous neighbouring cells. Clearly, contiguous cells have two
vertices in common.

My question: For a given row in TR, let us say row n, how best to find
all the other rows in TR which share two elements with row n? (The
algorithm must of course reject row n in its resulting list). I have
a rather inefficient solution, by far the slowest part of my code, and
I'm desperate to increase its speed (hopefully ten fold). My routine
uses one FOR loop (over cells n) within which are many WHERE commands
(which search for neighbour candidates).

My thoughts: Is there a clever way of solving the problem using subtle
IDL techniques/routines? Speed is key. Can it be entirely
vectorized? I even thought VORONOI (being so fast and triangulate's so-
called 'dual') might yield some helpful clues (apparently not).
TRIANGULATE's connectivity list seems to be more of a distraction than
a help, since I'm after neighbouring cells not neighbouring generator
points.

A replacement solution would be a serious contribution to my research
(and of course would be acknowledged upon publication of a paper), and
I would be happy to do speed comparisons if desired.

Yours.