Subject: Re: TRIANGULATE. Finding contiguous cells efficiently?
Posted by Libertan on Tue, 27 Feb 2007 19:44:13 GMT
View Forum Message <> Reply to Message

Wox,

Okay, here's an code which surpasses my expectations; it seems to be over 10,000% faster. I kid ye not. initially vectorized, ends with simple loop over uncostly operations. I thought the invocation of SORT would be slow, but imagine it's written in C. would I be right in thinking that the overall trick is to use 1) as few operations as possible, 2) use vectorized forms, and 3) use IDL instrinsics written in C?

I'm sure it's not particularly well written, but what do you think? Writing fast codes in IDL is a tricky business. Wox, if you care and are prepared to email me your name, I'll acknowledge our discussion when/if I publish future results (in Astrophysical Journal).

```
x=randomu(seed,np)
y=randomu(seed,np)
TRIANGULATE, X, Y, trang
s=size(trang,/dimensions)
ntr=s(1)
trang=trang+!pi  ;perhaps unnecessary, but ensures all values are
greater than 1. see below.

 ;============================================================ =====
;
;Crux: make array of edges, instead of pairs of vertices. Using the
two vertices of each edge
;create a *single* unique 'value' for the edge.  Use fast SORT to find
pairs of like edges.  Address
; simlarly sorted listr of cells to solve problem. Here I have
(foolishly) chosen to add the logs
; of the two vertices; Integer-> real.  yields unique value for the
edge?
; Instead, would something like x+1./y guarantee uniqueness?
;uniqueness of value ultimately limited by numerical precision?
 ;============================================================ =====

tred=dblarr(3,ntr)
tred(0,*)=double(alog(trang(0,*)))*double(alog(trang(1,*)))  ;need
arguments >1 ideally. see above
tred(1,*)=double(alog(trang(1,*)))*double(alog(trang(2,*)))  ; think
of better operation.
tred(2,*)=double(alog(trang(2,*)))*double(alog(trang(0,*)))
```

```
numtred=LONG(n_elements(tred))   ;=3*ntr

 ;============================================================ =====
; turn into vector of edges, instead of array, sort.
 ;============================================================ =====

edgvec=reform(tred,numtred)
celvec=LONG(findgen(LONG(3.*ntr))/3.)  ; =(0,0,0,1,1,1,2,2,2, etc.
Vector of cells associated
; with edgvec

edgsort=LONG(sort(edgvec, /L64))   ; sort order of edges
edgvecs=edgvec(edgsort)    ; sorted into edge value order
celvecs=celvec(edgsort)   ; likewise rearrange cells to keep track of
edge-cell relationship

;print, 'edgvec  :',edgvec
;print, 'celvec  :',celvec
;print, 'Now sorted...'
;print, 'edgvecs :',edgvecs
;print, 'celvecs :',celvecs

 ;============================================================ =====
; check that edges have unique values? skip.
 ;============================================================ =====

neigh=intarr(3,ntr)
neigh(*,*)=-1       ; any cells on convex hull will have one unpaired
edge -> -1

nedgvec=n_elements(edgvecs)
edgcount=LONG(intarr(ntr))

print, 'Starting loop over edges'

for i=0L, LONG(nedgvec-2) do begin   ; loop over edges, ordered by
their unique values.

       cellhere=LONG(celvecs(i))
       cellnext=LONG(celvecs(i+1))

       if(edgvecs(i) eq edgvecs(i+1)) then begin

              neigh(edgcount(cellhere),cellhere)=cellnext  ;this
cell

neigh(edgcount(cellnext),cellnext)=cellhere  ;complementary cell
```

```
            edgcount(cellhere)=edgcount(cellhere)+1L
            edgcount(cellnext)=edgcount(cellnext)+1L

        endif

endfor

end
```