Subject: Re: Looking for tetrahedra. Searching sorted lists.
Posted by cgguido on Tue, 16 Aug 2005 02:53:32 GMT
View Forum Message <> Reply to Message

Hi Karl,

thanks for replying.

> One thing worth pointing out is that nearly all mesh-related functions
> in IDL are implemented in C because those algorithms call for a lot of
> looping and other operations that are not efficient in IDL.

Am not sure what you mean by mesh-related... my data comes from the 3D
positions of ~5000 brownian particles diffusing around (each tagged
with an id). I can determine whether particles are nearest neighbours
two by two (which gives me the input matrix I mention in the original
post) and now am looking for quadruplets of mutually nearest neighbours
particles...

> Since you've got an algorithm implemented in IDL and an apparent need to
> run it repeatedly on lots of data that takes hours to process with IDL,
> this may be a good example of something to translate to C and package up
> as a DLM.

That's a possibility I would have never thought of, thanks! Although I
extensively use 'where' and 'uniq' and I don't know if they exist in C.
(at the very least, they must be available as a library...)

> You might post the IDL code to see if anyone can spot any obvious
> "low-hanging fruit". There may be some simple operations that you are
> doing that are very expensive and can be addressed without understanding
> or changing the main algorithm. For example, how are you collecting the
> quads? If you are just appending them to an array to create a new array,
> that can be costly.

Code appended. Yeah, I originally was appending each line one at a
time... got rid of that pretty fast! :-)

> I also can't understand your problem well enough to suggest that IDL code
> might be able to do it fast enough. The code might better describe what
> you are doing.

Am not sure what else to say to explain the problem, do you have
questions? The header on the code might help too...

Many thanks again, and appologies for the long code that follows.

Here is the code:

```
 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;;;;;;;;;;;
;
; NAME:
;  TETRAHEDRA
;
; WARNING:
;  Input must be sorted and each entry unique! See below.
;+
; PURPOSE:
;  Calculates all possible tetrahedra formed by nearest neighbours.
;  Tetrahedra are defined as quadruplets (obviously) of mutually
;  nearest-neighbour particles.
;-
; INPUT:
;  Uses the output of urstrtri.pro which is formated as follows:
;  [id0, id1, zero, timestamp]  (with id0 < id1). Each line is unique!
;
; PARAMS:
;  VERBOSE: Set /verbose to print out usefull information during
;  execution.
;  CHRONO: Set /chrono to print out duration of calculations in
seconds.
;  DEBUG:  Set /debug to only run the function on the first time stamp
;  and the first reference particle. You can then manually check that
;  *all* triangles were found.
;
; OUTPUT:
;  Outputs an array with one line per tetrahedron formatted as
;  follows:
;    [id0, id1, id2, id3, timestamp]
;  The list is sorted in ascending id order along both dimensions.
;  Outputs [-1,-1,-1,-1] if no tetrahedra are found.
;
; EXAMPLE:
;  To get all tetrahedra formed by particles in 'tri' and store the
;  output in 'a' while printing out some information about what's
;  going on and while timing the execution type:
;    a = tetrahedra(tri, /verbose, /chrono)
;
; HISTORY:
;  7-26-04 Written by Gianguido Cianci
;  7-28-04 Gianguido Cianci
;          - changed how the list of tetrahedra is created and
;            updated.
;          - added chrono option to measure times of execution.
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;;;;;;;;;;;

FUNCTION tetrahedra, tri, verbose=verbose, debug=debug, $
chrono=chrono, countbonds=countbonds

Compile_Opt IDL2

;;initialise timer
IF keyword_set(chrono) THEN BEGIN
   t0=systime(/seconds)
   currentt=t0
ENDIF

;find list of unique time stamps in tri
ut=uniq(tri[3,*])
utimes=tri[3,ut]
tmax=n_elements(ut[*])

IF keyword_set(debug) THEN tmax = 1

;;initialise tetrahedra list and tetrahedra counters
a = intarr(5,50000L*tmax)
row = 0L
old_row = row

;loop over time stamps
FOR tstampi = 0,tmax-1 DO BEGIN

   tstamp=utimes[tstampi]
   IF keyword_set(verbose) THEN $
   message, /inf,' Doing time stamp
'+strcompress(tstampi+1)+'/'+strcompress(tmax)

   ;;get all neighbours at t = tstamp
   wt = where(tri[3,*] EQ tstamp)
   trt = tri[*,wt]

   ;; get index id -> tri row
   trindex=i2p(trt, /track)
   ;;find list of particles with at least one neighbour
   u = uniq(trt[0,*])
   npart = n_elements(u)
   refid = trt[0,u]

   ;;debug using first particle with neighbours
   IF keyword_set(debug) THEN npart = 1
   ;;loop over particles with neighbours
   FOR i1 = 0,npart-1 DO BEGIN

      IF (keyword_set(verbose)) THEN BEGIN
         IF ((i1 MOD 500) EQ 0) THEN $
         message, /inf, '    Doing particle '+strcompress(i1+1) +' /'+
$
                  strcompress(npart)
      ENDIF

      ;;chose a particle
      p1 = refid[i1]

      ;;get its neighbours with id# > p1
      n1 = idneighbors(p1,trt, /gr)
      ;;;message, /inf, 'n1'+string(p1) & message, /inf, n1 ;debugging
stuff
      nneighbour = n_elements(n1)
      ;;if the ref particle has at least 3 neighbours then continue
      IF (nneighbour GT 2) THEN BEGIN

         ;;loop over neighbours of p1
         FOR i2 = 0,nneighbour-1 DO BEGIN

            ;;chose a neighbour of p1
            p2 = n1[i2]

            ;;find its neighbours with id# > p2
            n2 = idneighbors(p2, trt, /gr)
            ;;;message, /inf, 'n2'+string(p2) & message, /inf, n2
;debugging stuff

            ;;if second particle has at least 2 neighbours then
continue
            IF (n_elements(n2) GT 1) THEN BEGIN
               ;;concatentate lists of neighbours and sort them
               n3 = [n1,n2]
               n3 = n3[sort(n3)]

               ;;find id#'s that appear twice (i.e. look for triangles)
               u = uniq(n3)
               su = shift(u,1)
               du = u-su
               w = where(du[*] EQ 2)

               ;;if p1 and p2 form at least one triangle
               IF (w[0] NE -1) THEN BEGIN
                  jmax=n_elements(w)-1
            ;;loop over triangles p1 p2 p3
            FOR j = 0,jmax DO BEGIN

              ;;third particle in triangle p1 p2 p3
              p3 = n3[u[w[j]]]

              ;;it's neighbours with id# > p3
              n4 = idneighbors(p3, trt, /gr)

              ;;concatenate lists of neighbours and sort
              n5 = [n1,n2,n4]
              n5 = n5[sort(n5)]

              ;;find id#'s that appear thrice (i.e. look for
tetrahedra)
              uu = uniq(n5)
              suu = shift(uu,1)
              duu = uu-suu
              ww = where(duu[*] EQ 3)

              ;;loop over tetrahedra
              IF (ww[0] NE -1) THEN BEGIN

                kmax = n_elements(ww)-1
                FOR k = 0,kmax DO BEGIN

                  ;;fourth particle in tetrahedron p1 p2 p3 p4
                  p4 = n5[uu[ww[k]]]

                  ;;update list of tetrahedra
                  temp = [p1,p2,p3,p4,tstamp]
                  a[0,row] = temp
                  row = row + 1

                  ;; Using third col of tri to keep count
                  ;; nearest neighbour pair usage
                  IF (keyword_set(countbonds) ) THEN BEGIN
                    tri[2,wt[trindex[p1,p2]]]++

                    tri[2,wt[trindex[p1,p3]]]++

                    tri[2,wt[trindex[p1,p4]]]++

                    tri[2,wt[trindex[p2,p3]]]++

                    tri[2,wt[trindex[p2,p4]]]++
          tri[2,wt[trindex[p3,p4]]]++


         ENDIF ;END loop to count tetrahedral bonds
        ENDFOR   ;END loop over tetrahedra p1 p2 p3 p4
       ENDIF      ;END if p1, p2 and p3 form at least
one tetrahedron
      ENDFOR       ;END loop over triangles p1 p2 p3
     ENDIF          ;END if p1 and p2 form at least one
triangle
    ENDIF              ;END if second particle has at least 2
neighbours
   ENDFOR               ;END loop over neighbours of p1
  ENDIF                  ;END if the ref particle has at least
3 neighbours
 ENDFOR                   ;END loop over particles with
neighbours

 ;;Print info after each stack is analyzed.
 IF keyword_set(chrono) THEN BEGIN
  dt=ceil(systime(/seconds)-currentt)
  message, /inf, strcompress(dt)+ ' seconds elapsed for this
stack.'
  currentt=systime(/seconds)
  message, /inf, strcompress(ceil(currentt-t0))+ $
     ' seconds elapsed since the beginning of analysis.'
 ENDIF
 message, /inf, strcompress(row-old_row)+' tetrahedra found in this
stack.'
 message, /inf, strcompress(row)+' tetrahedra found overall, this
far.'
 message, /inf, ' --'
 old_row = row
ENDFOR                        ;END loop over time stamps


;count tetrahedra
ntetrahedra = row


message, /inf, ' --'
message, /inf, 'Found '+strcompress(ntetrahedra)+ ' tetrahedra.'

IF keyword_set(chrono) THEN BEGIN
 t0=ceil(systime(/seconds)-t0)
 message, /inf, 'Time taken: ' +strcompress(t0)+ ' seconds.'
ENDIF
```

```
;if no tetrahedron are found return [-1,-1,-1,-1,-1]
IF (ntetrahedra EQ 0) THEN BEGIN
   a = [-1, -1, -1, -1, -1]
   return, a
ENDIF

return, a[*,0:ntetrahedra-1]

END;; End of tetrahedra()
```