```
pro results_voronoi_volume, regions, q, volume=volume

if (q EQ !null) then q = lindgen(n_elements(*regions.volume))

for i=0,n_elements(q)-1 do $
  if ((*regions.volume)[q[i]] EQ 0) then begin
    vv = *regions.vertices[q[i]]

    hullfile = ('hull' + strint(round(random_nr(1)*1000000)) + '.dat')[0]
    openw, lun, hullfile, /get_lun
    printf, lun, '3'
    printf, lun, n_elements(vv)/3
    printf, lun, transpose(vv)
    free_lun, lun

    spawn, ['qconvex', 's', 'FS', 'TI', hullfile], out, ss, /noshell
    out = out[1]
    (*regions.volume)[q[i]] = double((strsplit(out, /extract))[2])
    if ((*regions.volume)[q[i]] LE 0) then stop

    spawn, 'rm ' + hullfile
  endif

end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

function results_voronoi, output

tstart = systime(1)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Computes the Voronoi connectivity for a set of points
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

pts = [[*output.x], [*output.y], [*output.z]]

;; Save the points to a temporary file
sz = size(pts)

ptsfile = ('pts' + strint(round(random_nr(1)*1000000)) + '.dat')[0]
openw, lun, ptsfile, /get_lun
printf, lun, sz[2]
printf, lun, sz[1]
printf, lun, transpose(pts)
free_lun, lun

;; Compute the voronoi regions
```

```
52  spawn, ['qvoronoi', 's', 'p', 'FN', 'TI', ptsfile], out, ss, /noshell
53  spawn, 'rm ' + ptsfile
54
55  dim = long(out[0]) & nvert = long(out[1])
56  vertstring = out[2:2+nvert-1]
57  vertices = fltarr(nvert, dim)
58  for i=0L,nvert-1 do vertices[i,*] = float(strsplit(vertstring[i], /extract))
59
60  ct = 2+nvert
61  nreg = long(out[ct])
62  reg = out[ct+1:*]
63  if (n_elements(reg) NE nreg) then stop
64  if (nreg NE sz[1]) then stop
65
66  ;; Find the voronoi region for each packet
67  regions = {vertices:ptrarr(nreg, /allocate), volume:ptr_new(dblarr(nreg))}
68  for i=0,nreg-1 do begin
69    w = long(strsplit(reg[i], /extract))
70    if (n_elements(w) GT 1) then begin
71      w = w[1:*]
72      q = where(w LT 0, onedge)
73    endif else onedge = 1
74
75    ;; if there are negative indices, at edge of region, set Volume=infinite
76    if (onedge GT 0) then begin
77      *regions.vertices[i] = -1
78      (*regions.volume)[i] = 1e30
79    endif else *regions.vertices[i] = vertices[w,*]
80  endfor
81  tend = systime(1)
82  print, 'Results_voronoi: ', tend-tstart
83
84  return, regions
85
86  end
```