```
function gravity, loc, geometry, options, which

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Computes gravitational acceleration
;;
;; Equations of motion:
;;   dvxdt = sum_objects (GM * (x-x_obj))/(r_obj)^3
;;   dvydt = sum_objects (GM * (y-y_obj))/(r_obj)^3
;;   dvzdt = sum_objects (GM * (z-z_obj))/(r_obj)^3
;;     -- r_obj = sqrt( (x-x_obj)^2 + (y-y_obj)^2 + (z-z_obj)^2 )
;;     -- radiation pressure only valid for Na
;;   dndt = instantaneous change in density
;;
;;  Version History
;;  3.0 - 7/20/2010
;;    * Small upgrade
;;  2.0 - 10/22/08
;;    * Created file from accel.pro
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

common constants

n = (size(*loc.x))[1]
ct = n_elements(which) ;; number of objects

;; Determine positions of satellites for each packet
if (options.motion) $
  then locmoon, *loc.t, (*geometry.phi)[which], (*SystemConsts.a)[which], $
    (*SystemConsts.orbrate)[which], x=xsat, y=ysat, z=zsat $
  else locmoon, fltarr(n_elements(*loc.t)), (*geometry.phi)[which], $
    (*SystemConsts.a)[which], (*SystemConsts.orbrate)[which], x=xsat, y=ysat, z=zsat

;; Compute distances between packets and satellites
ii = replicate(1., ct)
jj = replicate(1., n)

xdiff = (*loc.x)[*,0]#ii - xsat
ydiff = (*loc.x)[*,1]#ii - ysat
zdiff = (*loc.x)[*,2]#ii - zsat
r3 = (xdiff^2 + ydiff^2 + zdiff^2)^1.5

;; Compute gravitational acceleration
GM = jj # (*SystemConsts.GM)[which]
ax = GM * xdiff/r3
ay = GM * ydiff/r3
az = GM * zdiff/r3

if (ct NE 1) then begin
  ax = total(ax, 2)
```

```
    ay = total(ay, 2)
    az = total(az, 2)
  endif

  ; Final resutls
  accel = dblarr(n,3)
  accel[*,0] = ax
  accel[*,1] = ay
  accel[*,2] = az

  return, accel

  end

```