```
pro add_perturbation, startloc, PerturbVel, options, seed

common constants

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Adds a perturbation to a pre-existing velocity distribution
;;
;; Version History
;;   2.0: created 10/24/08
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

npack = options.packets
case (PerturbVel.type) of
  'none':
  'gaussian': if (PerturbVel.sigma EQ 0) $
    then vperturb = replicate(PerturbVel.vprob, npack) $
    else begin
      maxv = PerturbVel.vprob + 4*PerturbVel.sigma
      velocity = findgen(1001)/1000.*maxv
      velocity = velocity[where(velocity GT PerturbVel.vprob - 4*PerturbVel.sigma)]
      f_v = GaussianDist(velocity, PerturbVel.vprob, PerturbVel.sigma)
      vperturb = MonteCarloDistribution(velocity, f_v, npack)

      ;; Choose the altitude -- f(alt) = cos(alt)
      altitude = dindgen(1001)/1000. * ((PerturbVel.altitude)[1]-$
        (PerturbVel.altitude)[0]) + (PerturbVel.altitude)[0]
      f_alt = cos(altitude)
      alt = MonteCarloDistribution(altitude, f_alt, npack)

      ;; Choose the aziumth
      if ((PerturbVel.azimuth)[0] GT (PerturbVel.azimuth)[1]) $
        then m = [(PerturbVel.azimuth)[0], (PerturbVel.azimuth)[1]+2*!pi] $
        else m = PerturbVel.azimuth
      az = (m[0] + (m[1]-m[0]) * random_nr(seed=seed, npack)) mod (2*!pi)

      *startloc.altitude = alt
      *startloc.azimuth = az

      v_north = sin(alt)
      v_corot = -cos(alt) * cos(az)
      v_rad = cos(alt) * sin(az)

      vxperturb = v_rad * vperturb
      vyperturb = v_corot * vperturb
      vzperturb = v_north * vperturb

      ;; Need to rotate the perturbation vectors to proper orientation
      ;; Want az=0 => corotational direction
      ;;      az=90 => radial direction
```

```

        ;; Starting velocity
        *startloc.vx += vxperturb
        *startloc.vy += vyperturb
        *startloc.vz += vzperturb
      endelse
    'trigaussian': begin
      if (PerturbVel.vxsigma EQ 0) $
        then vxperturb = replicate(PerturbVel.vxprob, npack) $
        else begin
          maxv = PerturbVel.vxprob + 4*PerturbVel.vxsigma
          velocity = (findgen(2001)/1000.-1)*maxv
          f_v = GaussianDist(velocity, PerturbVel.vxprob, PerturbVel.vxsigma)
          vxperturb = -MonteCarloDistribution(velocity, f_v, npack)
        endelse

      if (PerturbVel.vysigma EQ 0) $
        then vyperturb = replicate(PerturbVel.vyprob, npack) $
        else begin
          maxv = PerturbVel.vyprob + 4*PerturbVel.vysigma
          velocity = (findgen(2001)/1000.-1)*maxv
          f_v = GaussianDist(velocity, PerturbVel.vyprob, PerturbVel.vysigma)
          vyperturb = MonteCarloDistribution(velocity, f_v, npack)
        endelse

      if (PerturbVel.vzsigma EQ 0) $
        then vzperturb = replicate(PerturbVel.vzprob, npack) $
        else begin
          maxv = PerturbVel.vzprob + 4*PerturbVel.vzsigma
          velocity = (findgen(2001)/1000.-1)*maxv
          f_v = GaussianDist(velocity, PerturbVel.vzprob, PerturbVel.vzsigma)
          vzperturb = MonteCarloDistribution(velocity, f_v, npack)
        endelse

        ;; Need to rotate to the location of the packets
        ang = atan(-*startloc.x, *startloc.y)
        vxpert2 = vxperturb * cos(ang) - vyperturb * sin(ang)
        vypert2 = vxperturb * sin(ang) + vyperturb * cos(ang)
        vzpert2 = vzperturb

        ;; Starting velocity
        *startloc.vx += vxpert2/SystemConsts.rplan
        *startloc.vy += vypert2/SystemConsts.rplan
        *startloc.vz += vzpert2/SystemConsts.rplan
        end
    'sputtering': stop
    'charge exchange': charge_exchange_perturbation, startloc, PerturbVel, options
  endcase

  end
```