```
pro show_veldist, proc_info, run_info, vrange=vrange, theo=theo, actual=actual, $
  display=display

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; display = 0 --> does not plot
;; display = 1 --> only plots theoretical distribution function
;; display = 2 --> Also plots a random distribution function for correct # of packets
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

SystemConstants, run_info.planet, systemconsts

if (n_elements(npack) NE 1) then npack = min([100000, run_info.packets])
if (n_elements(display) EQ 0) then display = 2

dv = 0.01
vrange = findgen(20000L)*dv
vrange = vrange[where(vrange LE 100.)] + dv

dodist = 0
case strlowcase(proc_info.speeddist) of
  'gaussian': begin
    a = [1., proc_info.vprob, proc_info.delv]
    theo = gauss(vrange, a)
    theo = theo * npack/total(theo)/dv

    bin = 0.01
    act = randomn(seed, npack)*proc_info.delV + proc_info.vprob
    actual = histw(act, min=0., max=max(vrange), bin=bin)/dv
    print, 'Gaussian speed distribution'
    print, 'P(v) ~ exp(-(v-v_prob)^2/(2*sigma^2))'
    print, '  v_prob = ' + strtrim(string(proc_info.vprob), 2) + ' km/s'
    print, '  sigma = ' + strtrim(string(proc_info.delv), 2) + ' km/s'
    print, '************'

    q = where(theo*dv GT .1, c)
    xr=[0,vrange[q[c-1]]]
    ylog = 1
    yr = [.1/dv, max(theo)]
    end
  'flat': begin
    q = where((vrange GE proc_info.vprob-proc_info.delV/2.) and $
      (vrange LE proc_info.vprob+proc_info.delV/2.))
    theo = fltarr(n_elements(vrange))
    theo[q] = 1.0
    theo = theo * npack/total(theo)/dv

    bin = 0.01
    act = random_nr(npack)*proc_info.delV + proc_info.vprob - proc_info.delV/2.
    actual = histw(act, min=0., max=max(vrange), bin=bin)/dv
    print, 'Flat speed distribution'
    print, 'v_prob-sigma/2. <= v <= v_prob+sigma/2.'
    print, '  v_prob = ' + strtrim(string(proc_info.vprob), 2) + ' km/s'
    print, '  sigma = ' + strtrim(string(proc_info.delv), 2) + ' km/s'
    print, '************'

    xr=[proc_info.vprob-proc_info.delv*1.5, proc_info.vprob+proc_info.delv*1.5]
    ylog = 0
    yr= [0,max(theo)]*1.1

    end
  'sputtering': begin
    theo = sputdist(vrange, proc_info.vprob, proc_info.alpha, proc_info.beta, $
      run_info.atom, v_b=v_b)
    theo = theo * npack/total(theo)/dv
    bin=.1
    dodist = 1
    print, 'Sputtering speed distribution'
    print, 'f(v) ~ v^(2*beta+1) / (v^2 + v_b^2)^alpha'
    print, '  alpha = ' + strtrim(string(proc_info.alpha),2)
    print, '  beta = ' + strtrim(string(proc_info.beta), 2)
    print, '  v_b = sqrt(2*U/m) = ' + strtrim(string(v_b), 2) + ' km/s'
    print, '  U = ' + strtrim(string(proc_info.vprob), 2) + ' eV'
    print, '************'
    end
  'exponential': begin
    ;theo = expflux(vrange, proc_info.temperature, run_info.atom, proc_info.beta, v_t=v_t)
    ;theo = theo * npack/total(theo)/dv
    theo = fltarr(n_elements(vrange))
    bin=.01
    dodist = 1
    if (n_elements(v_t) EQ 0) then v_t = 0.
    print, 'Exponential speed distribution'
    print, 'f(v) ~ v^beta * exp(-(v/v_t)^2)'
    print, '  beta = ' + strtrim(string(proc_info.beta), 2)
    print, '  v_t = sqrt(2kT/m) = ' + strtrim(string(v_t), 2) + ' km/s'
    print, '  T = ' + strtrim(string(proc_info.temperature), 2) + ' K'
    print, '************'
    end
  else: begin
    print, 'This option is not ready'
    return
    end
endcase

if (dodist) then begin
  if (run_info.mintrack GT 0) then begin
    nil = where(vrange LT run_info.mintrack, comp=notnil)
    proc_info.prodrate = proc_info.prodrate * total(theo[notnil])/total(theo)
    theo[nil] = 0.
  endif

  sumdist = theo
  n = n_elements(theo)
  for i=0,n-2 do sumdist[i+1] = sumdist[i+1] + sumdist[i]
  sumdist = sumdist/total(theo)
  t = random_nr(npack)
  act = interpol(vrange, sumdist, t)
  actual = histw(act, min=0, max=max(vrange), bin=bin)/bin

  q = where(theo*dv GT 1, c)
  xr=[0,vrange[q[c-1]]]
  ylog = 1
  yr = [.1/dv, max(theo)]
endif

q = where(strmatch(*systemconsts.objects, run_info.startpoint, /fold))
v_esc = sqrt(-2*(*systemconsts.GM)[q]/(*systemconsts.radius)[q])*systemconsts.rplan

if ((display EQ 1) or (display EQ 2)) then begin
  plot, vrange, theo, /xst, yr=yr, title='!17' + proc_info.speeddist + $
    ' Speed Distibution', xtit='Speed (km/s)', ytit='f(v) (km/s)!e-1!n', ylog=ylog, xr=xr
  if (display EQ 2) then oplot, findgen(n_elements(actual))*bin, actual, color=2
  oplot, [v_esc,v_esc], yr*[.01,100], color=5, linest=2
  xyouts, v_esc*1.1, yr[0]*2, 'v!desc!n', color=5
endif

destroy_constants, systemconsts

end
```