```
pro rk_5, loc, h, input, which, delta

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; This does a 5th order RK step and computes the error estimate
;; See Numerical Recipes, ch 17.2
;;
;; For each step:
;;   f_x = v
;;   f_v = a
;;   f_f = -f * ioniz
;;
;; Version History
;; * 3.1: 4/27/2011
;;     * cleaning up a bit and checking the error estimate
;; * 3.0: 7/21/10
;;     * Updating for new structure architecture
;; * 2.2: 4/26/10
;;     * The radiation pressure function only looks to see if the packet is in the
;;       planet shadow. Adding fix to check for moon shadow also
;; * 2.1: 4/26/10
;;     * xyz_to_magcoords and ionization_rate now determine whether the packet is
;;       shadowed by planets and moons. rk_5 is consistent with changes in those
;;       routines
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

common constants

hh = h # [1., 1., 1.]
hh2 = h^2 # [1., 1., 1.]

;; RK coefficients
c2 = 0.2d & c3 = 0.3d & c4 = 0.8d & c5 = 8./9.d & c6 = 1d & c7 = 1d

b1 = 35d/384d & b2 = 0d & b3 = 500d/1113d & b4 = 125d/192d & b5 = -2187d/6784d
b6 = 11d/84d & b7 = 0d
b1s = 5179d/57600d & b2s = 0d & b3s = 7571d/16695d & b4s = 393d/640d
  b5s = -92097d/339200d & b6s = 187d/2100d & b7s = 1d/40d
b1d = b1-b1s & b2d = b2-b2s & b3d = b3-b3s & b4d = b4-b4s & b5d = b5-b5s & b6d = b6-b6s
  b7d = b7-b7s

a21 = 0.2d
a31 = 3d/40d & a32 = 9d/40d
a41 = 44d/45d & a42 = -56d/15d & a43 = 32d/9d
a51 = 19372d/6561d & a52 = -25360d/2187d & a53 = 64448d/6561d & a54 = -212d/729d
a61 = 9017d/3168d & a62 = -355d/33d & a63 = 46732d/5247d & a64 = 49d/176d
  a65 = -5103d/18656d
a71 = b1 & a72 = b2 & a73 = b3 & a74 = b4 & a75 = b5 & a76 = b6

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

```
;; Step 1
;; k1 = h*f(x_n,y_n)
magcoord = xyz_to_magcoord(loc, input)
ioniz1 = ionization_rate(loc, input, magcoord)
a1 = accel(loc, input, magcoord, which)
magcoord = 0

k1x = hh**loc.v
k1v = hh**a1.dvdt
k1f = -h**loc.frac*ioniz1
a1 = 0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Step 2
loc2 = {t: ptr_new(*loc.t-c2*h), $
        x: ptr_new(*loc.x+a21*k1x), $
        v: ptr_new(*loc.v+a21*k1v), $
        frac: ptr_new(*loc.frac+a21*k1f)}

magcoord = xyz_to_magcoord(loc2, input)
ioniz2 = ionization_rate(loc2, input, magcoord)
a2 = accel(loc2, input, magcoord, which)
magcoord = 0

k2x = hh**loc2.v
k2v = hh**a2.dvdt
k2f = -h**loc2.frac*ioniz2
loc2 = 0 & a2 = 0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Step 3
loc3 = {t: ptr_new(*loc.t - c3*h), $
        x: ptr_new(*loc.x + a31*k1x + a32*k2x), $
        v: ptr_new(*loc.v + a31*k1v + a32*k2v), $
        frac: ptr_new(*loc.frac + a31*k1f + a32*k2f)}

magcoord = xyz_to_magcoord(loc3, input)
ioniz3 = ionization_rate(loc3, input, magcoord)
a3 = accel(loc3, input, magcoord, which)
magcoord = 0

k3x = hh**loc3.v
k3v = hh**a3.dvdt
k3f = -h**loc3.frac*ioniz3
loc3 = 0 & a3 = 0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Step 4
loc4 = {t: ptr_new(*loc.t - c4*h), $
        x: ptr_new(*loc.x + a41*k1x + a42*k2x + a43*k3x), $
        v: ptr_new(*loc.v + a41*k1v + a42*k2v + a43*k3v), $
```

```
103          frac: ptr_new(*loc.frac + a41*k1f + a42*k2f + a43*k3f)}
104
105   magcoord = xyz_to_magcoord(loc4, input)
106   ioniz4 = ionization_rate(loc4, input, magcoord)
107   a4 = accel(loc4, input, magcoord, which)
108   magcoord = 0
109
110   k4x = hh**loc4.v
111   k4v = hh**a4.dvdt
112   k4f = -h**loc4.frac*ioniz4
113   loc4 = 0 & a4 = 0
114
115   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
116   ;; Step 5
117   loc5 = {t: ptr_new(*loc.t - c5*h), $
118           x: ptr_new(*loc.x + a51*k1x + a52*k2x + a53*k3x + a54*k4x), $
119           v: ptr_new(*loc.v + a51*k1v + a52*k2v + a53*k3v + a54*k4v), $
120           frac: ptr_new(*loc.frac + a51*k1f + a52*k2f + a53*k3f + a54*k4f)}
121
122   magcoord = xyz_to_magcoord(loc5, input)
123   ioniz5 = ionization_rate(loc5, input, magcoord)
124   a5 = accel(loc5, input, magcoord, which)
125   magcoord = 0
126
127   k5x = hh**loc5.v
128   k5v = hh**a5.dvdt
129   k5f = -h**loc5.frac*ioniz5
130   loc5 = 0 & a5 = 0
131
132   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
133   ;; Step 6
134   loc6 = {t: ptr_new(*loc.t - c6*h), $
135           x: ptr_new(*loc.x + a61*k1x + a62*k2x + a63*k3x + a64*k4x + a65*k5x), $
136           v: ptr_new(*loc.v + a61*k1v + a62*k2v + a63*k3v + a64*k4v + a65*k5v), $
137           frac: ptr_new(*loc.frac + a61*k1f + a62*k2f + a63*k3f + a64*k4f + a65*k5f)}
138
139   magcoord = xyz_to_magcoord(loc6, input)
140   ioniz6 = ionization_rate(loc6, input, magcoord)
141   a6 = accel(loc6, input, magcoord, which)
142   magcoord = 0
143
144   k6x = hh**loc6.v
145   k6v = hh**a6.dvdt
146   k6f = -h**loc6.frac*ioniz6
147   loc6 = 0 & a6 = 0
148
149   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
150   ;; Step 7
151   loc7 = {t: ptr_new(*loc.t - c7*h), $
152           x: ptr_new(*loc.x + a71*k1x + a72*k2x + a73*k3x + a74*k4x + a75*k5x + a76*k6x), $
153           v: ptr_new(*loc.v + a71*k1v + a72*k2v + a73*k3v + a74*k4v + a75*k5v + a76*k6v), $
```

```
154           frac: ptr_new(*loc.frac + a71*k1f + a72*k2f + a73*k3f + a74*k4f + a75*k5f + $
155             a76*k6f)}
156
157   magcoord = xyz_to_magcoord(loc7, input)
158   ioniz7 = ionization_rate(loc7, input, magcoord)
159   a7 = accel(loc7, input, magcoord, which)
160   destroy_structure, magcoord
161
162   k7x = hh**loc7.v
163   k7v = hh**a7.dvdt
164   k7f = -h**loc7.frac*ioniz7
165   ptr_free, loc7.t, loc7.x, loc7.v, loc7.frac, a7.dvdt
166
167   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
168   ;; Step 7 -- Compute the result
169
170   *loc.t = *loc.t - h
171   *loc.x = *loc.x + b1*k1x + b2*k2x + b3*k3x + b4*k4x + b5*k5x + b6*k6x
172   *loc.v = *loc.v + b1*k1v + b2*k2v + b3*k3v + b4*k4v + b5*k5v + b6*k6v
173   *loc.frac = *loc.frac + b1*k1f + b2*k2f + b3*k3f + b4*k4f + b5*k5f + b6*k6f
174
175   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
176   ;; Step 8 -- Estimate the error
177   deltax = abs(b1d*k1x + b2d*k2x + b3d*k3x + b4d*k4x + b5d*k5x + b6d*k6x + b7d*k7x)
178   deltav = abs(b1d*k1v + b2d*k2v + b3d*k3v + b4d*k4v + b5d*k5v + b6d*k6v + b7d*k7v)
179   deltaf = abs(b1d*k1f + b2d*k2f + b3d*k3f + b4d*k4f + b5d*k5f + b6d*k6f + b7d*k7f)
180   delta = {x:ptr_new(deltax), v:ptr_new(deltav), frac:ptr_new(deltaf)}
181
182   end
```