```
function result_rkintegrate, x0temp, x1temp, output, regions, resolution=resolution

compile_opt idl2, hidden

;;**************************************************************************
;;
;;  Driver routine to run the 5th order RK integrator from Numerical
;;  Recipies, 3rd Ed.
;;
;;  x0,x1 should be nx3 arrays
;;
;;  Solve function:
;;    dN/dt = n(r(t)), r(t) = x0 + t*(x1-x0), N(t=0) = 0,
;;    integrate from t=0->t=1 or x0->x1
;;
;; Function returns integral over path.
;;
;; Written by Matthew Burger
;; Version 4.0: 3/24/2011
;;
;;***************************************************************

if (n_elements(resolution) EQ 0) then resolution = 1d-6

x0 = x0temp & x1 = x1temp
sz0 = size(x0) & sz1 = size(x1)
if ~(array_equal(sz0, sz1)) then stop
if (sz0[0] EQ 1) then begin
  x0 = transpose(x0)
  x1 = transpose(x1)
  sz0 = size(x0)
endif
npts = sz0[1]

h = replicate(0.1d, npts)       ;initial guess at best stepsize

;Set variables in preparation for iteration
count = 0L  ;; number of steps taken

;These control how quickly the stepsize is increased or decreased between iterations
safety = .95
shrink = -.25
grow = -.2

; -- don't use this right now - may want to change
;; yscale = scaling parameter for each variable

timeinc = resolution

;******************************************************************************
;Keep takeing R.K. steps until every packet has reached the time of "image taken"
```

```
;*******************************************************************************

t = dblarr(npts)
N = dblarr(npts)
v0 = x1-x0
tend = 1. ;; integrate from 0 to 1

t_step = dblarr(npts,10000)
N_step = dblarr(npts,10000)

if (regions EQ !null) then regions = results_voronoi(output)

tremain = tend - t
moretogo = where(tremain GT timeinc, ntogo)
done = (ntogo EQ 0)
while ~(done) do begin
  ;Now generate sub-arrays containing only the particles that are still being tracked

  trem = tremain[moretogo]
  t_sub = t[moretogo]
  x0_sub = x0[moretogo,*]
  v0_sub = v0[moretogo,*]
  N_sub = N[moretogo]
  h_sub = h[moretogo]

  ;Adjust stepsize to be no more than time remaining
  h_sub = (h_sub LE trem)*h_sub + (h_sub GT trem)*trem

  ;; Run the rk5 step
  result_rk5, t_sub, h_sub, x0_sub, v0_sub, N_sub, delta, output, regions

  ;; Do the error check
  ;; scale = a_tol + |y| * r_tol
  scale = resolution + abs(N_sub)*resolution

  ;; difference relative to acceptable difference
  delta /= scale

  ;; Check where difference is very small - adjust step size
  noerr = where(delta LE 1e-7)
  if (noerr[0] NE -1) then begin
    ;print, n_elements(noerr)
    delta[noerr] = 1.
    h_sub[noerr] = h_sub[noerr]*10.
  endif

  ;; Put the post-step values
  g = where(delta LE 1.0, ng, comp=b)
  if (ng GT 0) then begin
    t[moretogo[g]] = t_sub[g]
    N[moretogo[g]] = N_sub[g]
    h[moretogo[g]] = safety*h_sub[g]*delta[g]^grow
  endif

  if (ng NE ntogo) then begin
    ;; don't adjust the bad values, but do fix the stepsize
    htemp = safety * h_sub[b] * delta[b]^shrink
    q = where(htemp LT 0.0, nq) & if (nq NE 0) then stop

    ;; don't let step size drop below 1/10th previous step size
    h[moretogo[b]] = max([[htemp], [0.1*h_sub[b]]], dim=2)
  endif
  qqq = where(h LT 1e-7) & if (qqq[0] NE -1) then stop  ;; error test
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  tremain = tend - t
  moretogo = where(tremain GT timeinc, ntogo)  ;; check to see which ones aren't done
  count++ ;; step counter
  if (count mod 10 EQ 0) then print, 'Step Number: ' + strint(count) + $
    ', Points Remaining: ' + strint(ntogo)

  ;If it goes 100000 steps then it will never stop!
  done = ((ntogo EQ 0) or (count GT 100000.))
endwhile
q = where(N LT 0, nq) & if (nq GT 0) then stop

return, N

end
```