```
function extract_distribution, genericfiles, tempinput, firstfile=firstfile

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;  Takes a model output file with an isotropic surface distribution and flat
;;  speed distribution and creates a new outputfile with a specified speed and
;;  surface distribution
;;
;;  Required Inputs:
;;    genericfiles: list of generic output files
;;    input: new inputs structure
;;    firstfile: name of the first file to save (will increment automatically)
;;       If not given, then chooses file based on inputs
;;
;;  Version History:
;;    3.3: 7/8/2011
;;      * added PSD spatial distribution
;;    3.2: 4/27/2011
;;      * trackloss is now optional
;;    3.1: 1/6/2011
;;      * now extracts the distribution from a list of files and combines them
;;        to make a series of files with ~10^6 packets in each
;;    3.0: 7/16/2010
;;      * rewriting to make use of inputs structure
;;    2.1: 3/10/2010
;;      * allowing option of saving the results (which is slow) or just outputing them
;;          with keywords
;;    2.0: Created. 12/10/2009
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

common constants

if ~(file_test('tempoutput/')) then file_mkdir, 'tempoutput'

;; Remember these values for later
temptime = tempinput.options.endtime
temptaa = tempinput.geometry.taa

ngen = n_elements(genericfiles)
tempfiles = strarr(ngen)
packets = lonarr(ngen)
oldtaa = dblarr(ngen)
for i=0,ngen-1 do begin
  tempfiles[i] = 'tempoutput/temp' + strint(i) + '.output'

  ;; restore a generic file
  restore, genericfiles[i]
  oldinput = temporary(input)
  oldtaa[i] = oldinput.geometry.taa

  ;; This is the new input
```

```
    input = temporary(tempinput)

    ;; use the endtime and taa from the generic file
    input.options.endtime = oldinput.options.endtime
    input.geometry.taa = oldinput.geometry.taa

    ss = (where(*SystemConsts.objects EQ input.geometry.startpoint))[0]

    ;;;;;;;;;;;;;;;;
    ;; Determine the new surface distribution
    SpatialDist = input.SpatialDist
    if (ss EQ 0) then begin
      longitude = atan(*output.x0, -*output.y0)
      latitude = asin(*output.z0/SpatialDist.exobase)
    endif else begin
      longitude = atan(-*output.x0, -*output.y0)
      latitude = asin(*output.z0/SpatialDist.exobase)
    endelse
    longitude = (longitude + 2*!pi) mod (2*!pi)
    q = where(finite(longitude) EQ 0, nq) & if (nq NE 0) then stop
    q = where(finite(latitude) EQ 0, nq) & if (nq NE 0) then stop

    case strlowcase(SpatialDist.type) of
      'surface': begin
        case (1) of
          (SpatialDist.use_map): begin
            restore, SpatialDist.mapfile
            q = where(finite(*sourcemap.map) EQ 0, nq) & if (nq NE 0) then stop
            newf_spat = interpolate_xy(*sourcemap.map, *sourcemap.longitude, $
              *sourcemap.latitude, longitude, latitude)
            destroy_structure, sourcemap
            end
          (SpatialDist.longitude)[0] LE (SpatialDist.longitude)[1]: $
            newf_spat = ((longitude GE (SpatialDist.longitude)[0]) and $
              (longitude LE (SpatialDist.longitude)[1]) and $
              (latitude GT (SpatialDist.latitude)[0]) and $
              (latitude LE (SpatialDist.latitude)[1]))
          (SpatialDist.longitude)[0] GT (SpatialDist.longitude)[1]: $
            newf_spat = ((longitude GE (SpatialDist.longitude)[0]) or $
              (longitude LE (SpatialDist.longitude)[1])) and $
              (latitude GT (SpatialDist.latitude)[0]) and $
              (latitude LE (SpatialDist.latitude)[1])
          else: stop
        endcase
        end
      'psd': begin
        sourcemap = PSDfluxmap(input)
        *sourcemap.map /= max(*sourcemap.map)
        newf_spat = interpolate_xy(*sourcemap.map, *sourcemap.longitude, $
          *sourcemap.latitude, longitude, latitude)
        destroy_structure, sourcemap
```

```
103         end
104       else: stop
105     endcase
106     if (max(newf_spat) EQ 0) then stop
107     newf_spat /= max(newf_spat) ;; normalize to 1
108     q = where(newf_spat LT 0, nq) & if (nq GT 0) then newf_spat[q] = 0
109     q = where(finite(newf_spat) EQ 0, nq) & if (nq GT 0) then stop
110
111     ;;;;;;;;;;;;;;;;;;;;;;;;
112     ;; Determine the new speed distribution
113     vold = sqrt(*output.vx0^2 + *output.vy0^2 + *output.vz0^2)*SystemConsts.rplan
114     SpeedDist = input.SpeedDist
115     newf_vel = SpeedDistribution(input, vold)
116     if (max(newf_vel) EQ 0) then stop
117     q = where(newf_vel LT 0, nq) & if (nq GT 0) then newf_vel[q] = 0
118     q = where(finite(newf_vel) EQ 0, nq) & if (nq GT 0) then stop
119
120     ;;;;;;;;;;;;;;;;;;
121     ;; Determine the new angular distribution
122     AngularDist = input.AngularDist
123
124     zenang = (*output.x0**output.vx0 + *output.y0**output.vy0 + $
125       *output.z0**output.vz0)/sqrt(*output.x0^2+*output.y0^2+*output.z0^2)/$
126       sqrt(*output.vx0^2+*output.vy0^2+*output.vz0^2)
127     q = where(zenang LT 0L, nq) & if (nq GT 0) then zenang[q] = 0
128     q = where(zenang GT 1L, nq) & if (nq GT 0) then zenang[q] = 1
129     altitude = !pi/2-acos(zenang)
130     q = where(abs(altitude) GT !pi/2., nq) & if (nq GT 0) then stop
131
132     azimuth = fltarr(n_elements(*output.x0))
133
134     case (AngularDist.type) of
135       'radial': stop
136       'isotropic': newf_ang = ((altitude GE (AngularDist.altitude)[0]) and $
137         (altitude LE (AngularDist.altitude)[1]) and $
138         (azimuth GE (AngularDist.azimuth)[0]) and $
139         (azimuth LE (AngularDist.azimuth)[1]))
140       'costheta': begin
141         ;; tested this -- can reproduce costheta distribution in angular_distribution.pro
142         inrange = ((altitude GE (AngularDist.altitude)[0]) and $
143           (altitude LE (AngularDist.altitude)[1]) and $
144           (azimuth GE (AngularDist.azimuth)[0]) and $
145           (azimuth LE (AngularDist.azimuth)[1]))
146         newf_ang = sin(altitude)^AngularDist.n * inrange
147         end
148     endcase
149     if (max(newf_ang) EQ 0) then stop
150     newf_ang /= max(newf_ang)
151     q = where(newf_ang LT 0, nq) & if (nq GT 0) then newf_ang[q] = 0
152     q = where(finite(newf_ang) EQ 0, nq) & if (nq GT 0) then stop
153
```

```
154     weight = float(newf_vel*newf_spat*newf_ang)
155     if (max(weight) EQ 0) then stop
156     weight/= max(weight)
157     q = where(weight LT 0, nq) & if (nq GT 0) then stop
158     q = where(finite(weight) EQ 0, nq) & if (nq GT 0) then stop
159
160     *output.f0 *= weight
161     *output.frac *= weight
162     q = where(*output.f0 GE 1e-6, nq)
163     packets[i] = nq
164
165     *output.x0 = (*output.x0)[q]
166     *output.y0 = (*output.y0)[q]
167     *output.z0 = (*output.z0)[q]
168     *output.f0 = (*output.f0)[q]
169     *output.vx0 = (*output.vx0)[q]
170     *output.vy0 = (*output.vy0)[q]
171     *output.vz0 = (*output.vz0)[q]
172     *output.phi0= (*output.phi0)[q]
173     *output.time = (*output.time)[q]
174     *output.x = (*output.x)[q]
175     *output.y = (*output.y)[q]
176     *output.z = (*output.z)[q]
177     *output.frac = (*output.frac)[q]
178     *output.vx = (*output.vx)[q]
179     *output.vy = (*output.vy)[q]
180     *output.vz = (*output.vz)[q]
181     *output.sourcefile = genericfiles[i]
182     output.totalsource = total(double(*output.f0))
183
184     if (input.options.trackloss) then begin
185       *output.lossfrac = (*output.lossfrac)[q]
186       *output.ringfrac = (*output.ringfrac)[q]
187       *output.leftfrac = (*output.leftfrac)[q]
188       s = (size(*output.hitfrac))[0]
189       if (s EQ 1) $
190         then *output.hitfrac = (*output.hitfrac)[q] $
191         else *output.hitfrac = (*output.hitfrac)[q,*]
192       *output.deposition.longitude = 0
193       *output.deposition.latitude = 0
194       *output.deposition.map = 0
195     endif
196
197     ;; Note -- not saving deposition
198     save, input, output, version, file=tempfiles[i]
199
200     destroy_structure, output
201     tempinput = temporary(input)
202     tempinput.geometry.taa = temptaa
203     tempinput.options.endtime = temptime
204     print, 'extracted file ' + strint(i+1) + ' of ' + strint(ngen)
```

```
205  endfor
206
207  ;; Now go through and combine files to get ~10^6 packets per file
208  totpack = packets
209  i = 0 & cur = 0
210  curpack = 0L & maxpack = 1000000L
211  fnumber = lonarr(ngen)
212  while (i LT ngen) do begin
213    curpack += packets[i]
214    if (curpack LE maxpack) then begin
215      fnumber[i] = cur
216      i++
217    endif else begin
218      cur++
219      curpack = 0L
220    endelse
221  endwhile
222
223  n = max(fnumber)
224  newpack = lonarr(n+1)
225  outputfiles = strarr(n+1)
226  for i=0,n do begin
227    tempinput.geometry.taa = oldtaa[i]
228    q = where(fnumber EQ i, nq)
229    if (firstfile EQ !null) $
230      then outputfiles[i] = output_filename(tempinput) $
231      else stop
232    combine_iterations, outputfiles[i], tempfiles[q]
233    print, 'Created file: ' + file_basename(outputfiles[i]) + ' out of ' + $
234      strint(nq) + ' smaller files.'
235    print, 'Number of packets = ' + strint(total(packets[q]))
236  endfor
237  tempinput.geometry.taa = temptaa
238
239  ;; Remove the temporary files
240  spawn, 'rm -r tempoutput'
241
242  return, outputfiles
243
244  end
```