Last updated: August 22, 2011

## Contents

## 1 produce_results.pro

This procedure is the starting point for creating results from model outputs. An input structure or file and a format structure or file are required. The modeloutput files associated with the input file should be created before running this routine.

The possible result types are:

- image = 2-D image from a specfied viewing geometry
- voronoi image = 2-D image using the voronoi regions method
- los = the line-of-sight results along specific spacecraft trajectory and look direction
- points = the density sampled at specified points or along a spacecraft trajectory

The quantities that can be determined are:

- Density
- Column density

- Intensity (multiple excitation processes possible)
- Spectrum (simulated spectrum over wavelength range)

The rest of this section outlines the produce_results procedure.

1 Function call: `produce_results, inputtemp, formattemp, data=data, npackets=npackets, savefile=savefile`

  1.1 `inputtemp` can be:

    1.1.1 an input file that can be restored

    1.1.2 an input structure

    1.1.3 list of output files to restore

      - The input structure is determined from the first output file in the list
      - If the input structure is not the same for each output file, the results will be unpredictable (i.e., wrong).

  1.2 `formattemp` can be:

    1.2.1 a format file that can be restored

    1.2.2 a format structure

  1.3 `data` = points at which to produce results

    1.3.1 if producing LOS results, data is a structure with (x, y, z, xbore, ybore, zbore)

    1.3.2 if producing density or column density, data is a structure with (x, y, z)

  1.4 `npackets` = minimum number of packets needed to produce a result (default = 0)

  1.5 `savefile` = file to save the loaded packets into - still working on this option

2 Common block *results* contains:

  2.1 input = the input structure for this model run

  2.2 format = the format structure for this result

  2.3 SystemConsts

  2.4 stuff

  2.5 gvalue

  2.6 plasma = current plasma parameters for the simulation

3 *stuff* structure contains:

  3.1 aplanet = distance of planet from the sun (AU)

  3.2 vr = radial velocity of planet relative to the sun (km/s)

  3.3 atoms_per_packet = number of atoms 1.0 packets represents

  3.4 mod_rate = rate packets are ejected from the surface

  3.5 totalsource = total number of packets ejected

4 Restore the inputs and determine which output files to use

5 Restore the SystemConsts structure and determine the distance and radial velocity of the central planet based on input.geometry.taa
- function calls to SystemConstants and planet_dist

6 Determine *totalpackets* = total number of packets available

   6.1 extracts the number of saved packets from the output file headers and

   6.2 If the *npackets* keyword is not set, then $npackets = 0$.

   6.3 If no output files are found, then $totalpackets = 0$

   6.4 If $totalpackets < npackets$ then there is nothing to do.

7 Restore the format structure if the filename is given as an input

8 Determine *stuff.totalsource* by extracting the values from the output files

   8.1 $totalsource = \sum output.frac0$ → total starting fractional values for the packets

   8.2 *totalpackets* is the number of packets ignoring the initial fractional value

9 $stuff.mod\_rate = stuff.totalsource/input.options.endtime$ → packets ejected per second

10 $stuff.atoms\_per\_packet = format.strength \times 10^{26}$ → number of atoms each packet represents. $fomat.strength$ = source rate in units of $10^{26}$ atoms/sec.

11 if $format.quantity$ = 'intensity', then setup the intensity parameters
- run procedure results_intensity_setup

12 Run the appropriate results production program based on format.type:

   12.1 'image': Create an image from a specified viewing geometry
   - result = produce_image(files, savefile=savefile)

   12.2 'voronoi image': Create an image from a specified viewing geometry using the voronoi region method (still in progress)
   - result = produce_voronoi_image(files, savefile=savefile)

   12.3 'los': Determine result along a trajectory line of sight
   - result = produce_los(files, savefile=savefile)

   12.4 'points': Determine density at specified points
   - result = produce_density(files, savefile=savefile)

## 2 Format Structures

## 3 results_intensity_setup

1 procedure is in the file *results_functions_4.0.pro*

2 This routine loads the gvalues and plamsa info for resonant scattering and electron impact excitation

3 Resonant scattering emission:

3.1 format.emission.mechanism must contain 'resscat'

3.2 g-value determined from get_gvalue, which gives the gvalues for specified atom at specified distance from the sun as function of radial velocity. The g-value structure contains:

3.2.1 species: the emitting atom

3.2.2 a: distance from sun (AU)

3.2.3 wavelength: array of emission line wavelengths (Å)

3.2.4 v: array of radial velocities relative to the sun (km $s^{-1}$)

3.2.5 g: $(n_v \times n_\lambda)$-array with g-value as function of $v_r$ and $\lambda$ (photons atom$^{-1}$ s$^{-1}$)

3.2.6 radaccel: radiation accelleration as function of $v_r$ (cm s$^{-2}$)

4 Electron impact excitation

4.1 format.emission.mechanism must contain 'eimp'

4.2 **This has not been set up correctly yet.**

# 4 produce_image

This procedure produces a 2-D image from a specified viewing geometry → See Section 2 for required parameters. The user does not call this function directly – it is called from produce_results.

The procedure is outlined below:

1 Determine the image origin – There are some differences in the calculations depending on whether the image center is the planet or a satellite.

2 Image dimensions:

2.1 Size of the output image (number of pixels) is given by `format.geometry.dims`

2.2 The image center is given by `format.geometry.center` measured in $R_{obj}$ relative to the center of the image origin (`format.geometry.origin`).

2.3 The width of the image is given by `format.geometry.width` measured in $R_{obj}$.

2.4 Note that `geometry.dims`, `geometry.width`, and `geometry.center` are all 2-element vectors.

3 Coordinate system

3.1 The packets are rotated into a reference frame with the x-axis aligned along the image horizontal axis, the z-axis aligned along the image vertical axis, and the y-axis along the image line of sight with the observer located at $(0, -\infty, 0)$

3.2 Image scale:

$$scale = \frac{geometry.width}{geometry.dims - 1} \tag{1}$$

## 5 produce_voronoi_image

## 6 produce_los

## 7 produce_points