Contents

1 Introduction

This document provides a guide to using Matthew Burger's neutral cloud model. You can contact me at Matthew.Burger@nasa.gov.

The document is organized as follows:

1 Setting things up: how to get the model running on your computer

2 Creating Packets

  2.1 Creating Input files

  2.2 Running the model

3 Displaying outputs

  3.1 Creating format files

  3.2 produce_results

4 What the model does: I try to explain the physical and computational processes in more detail and provide examples to prove it works the way it should.

5 Program Catalog: Help file and algorithm description for each program.

2 Setup

All of the files used for the model are listed in Appendix A. Everything necessary for running the model is synced to the LASP server `mascs_data` in the directory `Burger/NeutralModel/`. This contains the following directories:

1 `modelpro` → all the model code

2 `misc` → various helper functions including the IDL Astronomy Library and the ICY SPICE Library (SPICE for IDL)

3 `spice_kernels` → The SPICE kernels necessary for comparing model runs with MESSENGER data

4 `SummaryFiles` → The UVVS summary files created by Aimee Merkel.

Start IDL and compile the model by typing:

```
IDL> @PATH/compile_model_mercury
```

The routines *physical_constants* and textitPlanetaryConstants will set up a series of system variables with various physical constants and planetary data. The new system variables are: `!consts`, `!Sun`, `!Mercury`, ... `!Pluto`. To see what is contained in each variable, type:
`IDL> help, /struct,` *variable*

3 Input Files

The input file contains all the parameters needed to run the model. This should be all you need to change to get the output you are looking for. The file contains a series of lines in the format:
structure.field = value
String values should not have quote marks around them. Numerical values cannot be mathematical operations. For numerical values, it is not necessary to specify double precision. All the inputs are case insensitive. I use capitalization to make things clearer for me, but it is not necessary.

The inputs and outputs are contained in a series of structures. The input structures are:

1 geometry

2 sticking_info

3 forces

4 SpatialDist

5 SpeedDist

6 AngularDist

7 PerturbVel

8 plasma_info

9 options

This may seen like a lot of structures, but I've found that this allows a large number of options to be set while still keeping things organized. In the sections that follow, I descirbe each structure and give the possible options. Fields in red are optional. If a required field is not included in the input file, the run will crash. Appendix B summarizes the options.

3.1 geometry

The format of the geometry structure depends on the central object in the system.

1 Mercury

- 1.1 planet = Mercury
- 1.2 taa = true anomaly angle in radians
- 1.3 include = 1/0 depending on whether to include Mercury's gravity as a force. (Default = 1)

2 Earth

- 2.1 planet = Earth
- 2.2 StartPoint = Earth or Moon
- 2.3 phi = final heliocentric orbital position for the Earth and Moon. The Earth's position is currently ignored, although I need to add in seasonal effects. This line needs to be included either 1 or 2 times. If only included once, then it is assumed to be for the moon.
- 2.4 include = 1/0 depending on whether to include the gravity from the Earth and moon. If present, must be included twice. Default = 1 for each object.

3 Jupiter

- 3.1 planet = Jupiter
- 3.2 StartPoint = Jupiter or one of its moons
- 3.3 CML = central meridian longitude in radians
- 3.4 phi = final heliocentric orbital position for the planet and each moon. The line `geometry.phi = xxx` must be included 5 times (one time each for Jupiter, Io, Europa, Ganymede, and Callisto, in order). The orbital position for Jupiter is always 0, so this line doesn't matter, although it must be included.

3.5 include = 1/0 depending on whether to include the gravity from Jupiter and each moon as a force. This line must be included 5 times. Default = 1 for all objects.

4 Saturn

4.1 planet = Saturn

4.2 StartPoint = Saturn or one of its moons

4.3 phi = final heliocentric position for the planet and each moon. This line must be included 10 times (one time each for Saturn, Mimas, Enceladus, Tethys, Dione, Rhea, Titan, Hyperion, Iapetus, and Phoebe, in order).

4.4 include = 1/0 depending on whether to include gravity from each object. This line must be included 10 times. Default = 1 for each object.

Regarding the *include* option: In general it is only necessary to include the gravity from the central object and the starting satellite. Including more satellites has little effect on particle trajectories over short time scales and will slow things down since more computations are necessary.

3.2 sticking_info

This structure controls what happens when when a packet hits the surface. Currently the sticking coefficient is a constant, but I will add more complicated functions. If the sticking coefficeint equals 1, then the entire packet sticks to the surface and the integration is complete. If it is less than 1, the additional fields in *sticking_info* specify how the packet is reemitted from the surface.

1 StickCoef = a number between 0 and 1 to specify the fraction of the packet which sticks to the surface.

2 If StickCoef = 1, then nothing else is required.

3 If StickCoef < 1, then the following options are possible:

3.1 Thermal accommodation to the surface temperature. The packet is reemitted in a random direction at a speed between the incident speed and random speed based on the local surface temperature. The new velocity is $v_1 = av' + (1-a)v_0$, where v' is chosen from the Maxwellian disrtibution function $f_M(v, T)$.

- emitfn = Maxwellian
- Tsurf = surface temperature (K). If Tsurf=0, then the local surface temperature is computed as a function of solar zenith angle (only set up for Mercury). Otherwise, the surface temperature is assumed constant.
- accom_factor = a number between 0 and 1 indicating the amount of thermal accommodation (1 = fully accommodated).

3.2 Elastic scattering. The packet is reemitted in a random direction at the same speed it hit the surface with. This is equivalent to setting emitfn to Maxwellian and accom_factor = 0 (no accommodation).

- emitfn = elastic scattering

3.3 forces

This structure controls which forces acting on the packets are computed. The possible forces are gravity, radiation pressure, and the Lorentz force. The Lorentz force doesn't work right now, and in any case is irrelevant for neutrals. Any force not explicitly listed in the input file is turned off.

Right now I have g-values for C, Ca, Ca^+, H, He, K, Mg, Mg^+, Na, OH, O, and S. If radiation pressure is set for any other species, then the g-value is assumed to be 0.

- gravity = 1/0 to turn gravity on/off (default = 0)
- radpres = 1/0 to turn radiation pressure on/off (default = 0)
- lorentz = 1/0 to turn the Lorentz force on/off (default = 0)

3.4 SpatialDist

This determines the initial spatial distribution of the packets. Currently, the options are: surface, torus, exosphere, and SO_2 exosphere. The required fields for each of these options are different.

There is an option called *random_even* included in each of these distributions which used to toggle between randomly choosing packets with the specified parameters or evenly distributing them throughout the region. The flag is still there, but only the random option works. It turns out that evenly distributing packets over a sphere is difficult and not really very useful.

1 Surface distribution. This distributes packets randomly over the surface of a sphere. There are two ways to specify how packets are distributed: by specifying a longitude/latitude range, or by supplying a probibility distribution function map.

   To specify a longitude and latitude range, the parameters are:

   - use_map = 0
   - exobase = radius of the sphere in units of the starting point (satellite or planet) radius. Default = 1, for starting at the surface. Exobase < 1 will cause problems.
   - longitude0 = lower bound on lonigutde [radians]. Default = 0.
   - longitude1 = upper bound on longitude [radians]. Default = 6.2831853.
   - latitude0 = lower bound on latitude [radians]. Default = -1.5707963
   - latitude1 = upper bound on latitude [radians]. Default = 1.5707963

   The latitude range is $-\pi/2$ to $\pi/2$. The longitude range is 0 to 2π. To choose a sub-region that spans the 0 longitude, set longitude0 > longitude1. (Setting a negative value will most likely not work right - I haven't checked.)

   The probability distribution functions for longitude (λ) and latitude (μ) are determined by:

$$f_\lambda \propto constant, \lambda_0 \leq \lambda \leq \lambda_1 \quad (1)$$
$$f_\mu \propto cos(\mu), \mu_0 \leq \mu \leq \mu_1 \quad (2)$$

I need to verify that this works according to the coordinate systems defined in Section 6.1 below, which are centered on the subsolar point, in the case of planets, and the sub-planet point in the case of satellites. (Because the planetary coordinate systems are right-handed and the satellite systems are left-handed, I expect that there are currently some mistakes).

If you would like to specify a more complicated surface distribution, you need to provide a mapfile in a format which I will describe later.

2 PSD. This distributes packets randomly over the surface of a sphere with a probability that depends on the incident photon flux and the proton precipitation flux. The PSD flux at each point on the surface is given by

$$\Phi_{PSD}(\lambda,\mu) = min\left\{\Phi_{p-l},\Phi_{d-l}\right\} \quad (3)$$

$$\Phi_{p-l}(\lambda,\mu) = \left(\frac{\phi_{phot}(1\text{ AU})}{r^2}\right)\times\cos(SZA)\times\sigma_{PSD}(T(\lambda,\mu))\times c(\lambda,\mu)n(\lambda,\mu) \quad (4)$$

$$\Phi_{d-l}(\lambda,\mu) = \phi_{diff}(\lambda,\mu)\times\left(1+10^{-8}\kappa\phi_{precip}\right) \quad (5)$$

where λ,μ are the longitude and latitude measured relative to the subsolar point, Φ_{PSD} is the PSD flux from the surface, and Φ_{p-l} and Φ_{d-l} are the photon- and diffusion-limited fluxes respectively.

The photon-limited flux depends on the photon flux ϕ_{phot} at Mercury's heliocentric (r), the solar zenith angle (SZA), the temperature dependent PSD cross section (σ_{PSD}), the concentration of desorbed species (c), and the surface density of the regolith (n). While σ, c, and n likely all vary with over the surface due to temperature variations and local geology, we assume constant values of $\sigma = 3\times10^{-21}$ cm^{-2}, $c = 0.005$, and $n = 7.5\times10^{14}$ as used by [**?**]. More complicated functions will be implemented as needed. At 1 AU, the photon flux at energies greater than 4 eV (wavelength $<$ 310 nm), the threshold for PSD is $\sim 2.8\times10^{15}$ photons cm^{-2} s^{-1}, which we use as a baseline solar flux (this can potentially be parameterized if necessary). $\cos(SZA)$ is equivalent to $\cos\lambda\cos\mu$ (demonstrated on Section **??** below), and is zero on the nightside where there is no solar flux.

The diffusion-limited flux is the rate at which atoms naturally diffuse from the interior of regolith grains to their exteriors. The temperature and compositional dependencies are not known; we assume the value is constant. Previous estimates [**?**, **?**] have found that $\phi_{diff}\sim 10^6-10^7$. κ is the ion-enhanced diffusion effectiveness, a parameter which governs how effective precipitating ions are at increasing the diffusion of atoms to grain surfaces. The factor of 10^{-8} keeps $\kappa\sim 1$.

The PSD spatial distribution is normalized such that the total source rate can be determined by a linear fit to the data:

$$f_{PSD}\propto min\left\{1,\rho\times\left(1+10^{-8}\kappa\phi_{precip}\right)\right\} \quad (6)$$

where $\rho = \Phi_{p-l,ss}/\phi_{diff}$ is the ratio of the photon limited flux at the sub-solar point to the diffusion limited flux. The advantage of this construction is that the modeled source rate determines both the photon-limited and diffusion-limited flux; the disadvantage is that ρ must be specified *a priori* and a separate model run is needed for each value of ρ that may be chosen. The is no simple way around this as the total PSD flux is not a simple linear combination of the non-enhanced and ion-enhanced PSD fluxes.

mnum	SW conditions
0	inbound IMF, $n_i = 10$ cm^{-3}
1	inbound IMF, $n_i = 20$ cm^{-3}
2	inbound IMF, $n_i = 30$ cm^{-3}
3	outbound IMF, $n_i = 10$ cm^{-3}
4	outbound IMF, $n_i = 20$ cm^{-3}
5	outbound IMF, $n_i = 30$ cm^{-3}

Table 1: Values for *mnum* in function `precipitation_modelnum`

The total source rate is given by:

$$S = \int_\Omega \Phi_{PSD} d\Omega = \int_0^{2\pi} \int_{-\pi/2}^{\pi/2} \Phi_{PSD} \sin\mu d\mu d\lambda \tag{7}$$

The function `PSD_flux` does something useful to convert between fluxes and total source rates.

The required inputs for the PSD spatial distribution are:

- ρ = ratio of photon-limited and non-ion enhanced diffusion flux at the sub solar point
- κ = effectiveness of ion-enhanced diffusion
- modnum = Ion precipitation model calculated by Mehdi Benna.

The parameter modnum refers to one of the MHD model simulations computed by Mehdi Benna; the ion precipitation flux to the surface is calculated as a function of solar wind ion density and IMF B_x, B_y, and B_z. For each MESSENGER orbit, six ion precipitation scenarios are given corresponding to two IMF configurations (measured before MESSENGER's inbound passage through the bow shock and after the outbound passage) and low (10 cm^{-3}), medium (20 cm^{-3}), and high (30 cm^{-3}) solar wind densities. The function *precipitation_modelnum* gives the model number to use as a function of orbit and SW configuration; a complete listing is given in the file XXXXXX.

```
IDL> modnum = precipitation_modelnum(orbit, mnum)
```

where *mnum* refers to the conditions in Table 1.

3 Torus. This begins packets in a longitudinally symmetric torus in the equatorial plane of the central planet. The locations of packets are determined by:

$$x = (r_0 + a\cos\theta)\cos\phi \tag{8}$$
$$y = (r_0 + a\cos\theta)\sin\phi \tag{9}$$
$$z = b\sin\theta \tag{10}$$

where a, b, θ, and ϕ are random variables in the ranges:

$$0 \le a \le r_1 \tag{11}$$
$$0 \le b \le r_2 \tag{12}$$
$$0 \le \theta \le 2\pi \tag{13}$$
$$0 \le \phi \le 2\pi \tag{14}$$
$$\tag{15}$$

This produces a torus with a major axis r_0, and minor axes r_1 in the equatorial plane and r_2 out of the equatorial plane.

Right now this can only create a torus around the central planet, although we may want to change that later to simulate production from the Rhea ring, for example.

Parameters:

- radius0 = torus major axis, r_0 [planetary radii]
- radius1 = torus minor axis, r_1 [planetary radii]
- radius2 = torus minor axis, r_2 [planetary radii]

Add some figures showing that this works.

4 Exosphere distribution. This starts packets in a spherically symmetric exosphere centered on the starting planet or moon. The exosphere can drop off either exponentially or as a powerlaw. The radial components of the powerlaw and exponential exospheres have the form:

$$f_r \propto r^b; powerlaw \tag{16}$$
$$f_r \propto e^(- (r-1)/b; exponential \tag{17}$$

For a powerlaw exosphere, b is the powerlaw index. For an exponential exosphere, b is the scale height. The angular components are isotropically distributed:

$$f_\theta \propto \cos\theta \tag{18}$$
$$f_\phi \propto constant \tag{19}$$

To simulate production of atomic species by photodissociation of a molecular exosphere, there is an option (block_shadow) to only start packets in sunlight.

Parameters:

- exotype = exponential or powerlaw
- b = scale height [object radii] for the exponential or powerlaw index. The powerlaw index should be < 0.
- rmax = cutoff distance for exosphere [object radii]. Default = 10.
- block_shadow = 0/1 to allow/block packets to start in the geometric shadow. Default = 0.

5 SO_2 exosphere. This was sent to me by Vincent Dols to simulate production of O in Io's SO_2 exosphere.

$$f_\phi \propto e^{-(|\phi|-\pi)^2/1.07^2} \tag{20}$$
$$f_r \propto -474.8 + 148.7r + 200.7r^2 \times \frac{0.09}{(r_{peak}(\phi) - r^2)^2} + 0.09^2 \tag{21}$$
$$r_{peak} = 1.42 \times \left[1 + 0.05\left(\frac{\pi - |\phi|}{\pi/4}\right)^2\right] \tag{22}$$
$$f_\theta \propto \cos\theta \tag{23}$$

There are no parameters to set.

Need to add a figure showing what this looks like.

3.5 SpeedDist

Options for the SpeedDist.type are: Gaussian, trigaussian, sputtering, Maxwellian (thermal), flat (constant), circular orbits, dolsfunction, and user defined.

1 Gauusian distribution. The Gaussian speed distribution has the form:

$$f_v(v) \propto e^{-(v-v_{prob})^2/2/\sigma^2} \tag{24}$$

Parameters are:

- vprob = most probable velocity [km/s]
- sigma = Gaussian width [km/s]

2 Tri-gaussian distribution. This allows you to set a different Gaussian speed distribution in the x, y, and z directions. This needs to be tested. I am not sure what coordinate system is used.

Parameters are:

- vxprob = most probable speed along x axis [km/s]
- vxsigma = Gaussian width in x direction [km/s]
- vyprob = most probable speed along y axis [km/s]
- vysigma = Gaussian width in y direction [km/s]
- vzprob = most probable speed along z axis [km/s]
- vzsigma = Gaussian width in z direction [km/s]

3 Sputtering speed distribution. This uses a generic sputtering distribution in the form:

$$f_v(v) \propto \frac{v^{2\beta+1}}{(v^2+v_b^2)^\alpha} \tag{25}$$

$$v_b = \left(\frac{2U}{m_{atom}}\right)^{1/2} \tag{26}$$

This is equivalent to an energy distribution in the form:

$$f_E(E) \propto \frac{E^\beta}{(E+U)^\alpha} \tag{27}$$

Required parameters are:

- U = binding energy [eV]
- alpha = α parameter
- beta = β parameter

Typical values for several species are given in Table 2.

Table 2: Sputtering parameters.

Species	U (eV)	α	β
H_2O from ice	0.055	3	1
O_2 from ice	0.015	2	0
Na from rock	2	3	1

4 Maxwell-Boltzmann distribution. The Maxwellian distribution has the form:

$$f_v(v) \propto v^2 e^{-v^2/v_{th}^2} \tag{28}$$

$$v_{th} = \left(\frac{2k_b T_{surf}}{m_{atom}}\right)^{1/2} \tag{29}$$

The only field used is the temperature. If the temperature is set to a number greater than 0, then the surface is assumed to be constant at that temperature (in K). If temperature is set to zero, a model for the local surface temperature as a function of solar zenith angle is used.

Currently, the only body with a surface temperature model is Mercury. The temperature is as determined by [**?**]. On the dayside ($SZA \leq \pi/2$:

$$T_s(SZA) = T_0 + T_1 \times cos^{1/4} SZA \tag{30}$$

$$T_0 = 100 \tag{31}$$

$$T_1 = 600 + 125 \times (\cos(TAA) - 1)/2. \tag{32}$$

On the nightside, $T_s = 100$ K. I'm not exactly sure where the equation for T_1 comes from. It is probably chosen that that T_1 varies between 475 K and 600 K, and the temperature at the subsolar point varies between 575 K and 700 K, as suggested by [**?**] and [**?**]. I think this formulation will need to be improved.

5 Flat speed distribution. The distribution function is given by:

$$f(v) = constant, v_{prob} - \Delta v \leq v \leq v_{prob} + \Delta v \tag{33}$$

$$f(v) = 0, otherwise \tag{34}$$

Parameters:

- vprob = center of the distribution [km/s]
- delv = half-width of the distribution [km/s]

6 Circular orbits. This starts the packets in circular, Keplarian orbits around the central planet. It will not start packets in orbits around satellites, although it would not be hard to add this feature if desired. The circular orbits distribution was designed to be used with the torus spatial distribution. The initial direction of packets is in the $\hat{z} \times \hat{r}$ direction, where $\hat{z}$ points north and $\hat{r}$ points radially from the origin to the packet. There are currently no settable options for this speed (velocity, really) distribution.

The purturb_vel structure works well with circular orbits to add small velocity perturbations to stable orbits.

7 Dolsfunction. This is a speed distribution for the production O and S from SO_2 dissociation. It was provided to me from Vincent Dols, based on measurements by [**?**].

$$f_v(v) \propto v \times e^{-(E-d_0)^2/d_1^2} \quad (35)$$

$$E = \frac{1}{2} m_{atom} v^2 \quad (36)$$

Looking at this now, I'm not 100% sure I did this right. We will need to confirm that it works.

Parameters:

- dols0 = d_0 parameter.
- dols1 = d_1 parameter.

8 User defined. The user provides a structure with fields speeddist.v and speeddist.fv. The only parameter to set is distfile.

3.6 AngularDist

The possible angular distributions are radial, isotropic, and costheta. It is also possible to turn this off if the structure is not needed.

1 No angular distribution. The circular orbits speed distribution does not currently allow the user to specify an angular distribution. If one is set, it will be ignored.

2 Radial distribution. Packets are sent out radially from the center of the starting object.

3 Isotropic. Packets are sent out isotropically into a sphere or, if using the surface spatial distribution, the outward hemisphere.

The probability distribution functions for the azimuth angle ϕ and the altitude angle θ are determined by:

$$f_\phi \propto constant, 0 \leq \phi < 2\pi \quad (37)$$

$$f_\theta \propto cos\theta, (-\pi/2, 0) \leq \theta \leq \pi/2 \quad (38)$$

Parameters:

- altitude0 = lower limit on the altitude angle [radians]. Default = 0 for SpatialDist.type = surface; $-\pi/2$ otherwise.
- altitude1 = upper limit on the altitude angle [radians]. Default = $\pi/2$
- azimuth0 = lower limit on the azimuth range [radians].
- azimuth1 = upper limit on the azimuth range [radians].

Notes:

3.1 The altitude is defined relative to the vector from the center of the starting object to the packet. Altitude = 90° points radially outward. Altitude = 0° is tangent to the object surface at that point.

3.2 The azimuth should be defined such that azimuth = 0° is east and azimuth = 90° is north, but I have not confirmed that I have this correct.

3.3 The azimuth should be in the range (0,2π). To choose a subrange that includes 0, set azimuth0 > azimuth1.

4 costheta angular distribution. Sets the altitude angle distribution propotional to $\cos^n(\psi) = \cos^n(\pi/2 - \theta)$ where θ is the altitude angle (measured relative to the surface tangent) and ψ is the zentih angle (measured relative to the surface normal). The altitude distribution is determined by:

$$f_\theta \propto \sin^n(\theta) \tag{39}$$

One could argue that this is not a good distribution because it does not take into account the decrease in the size of a dϕ bin near the poles, and that there should be an additional factor of $\cos\theta$ in there. I have left this out because I wanted a function that is peaked normal to the surface. If I've done this wrong, I can change it (actually, it would just require uncommenting one line and commenting another).

Parameters:

- n = exponent on the distribution. Default = 1.
- altitude0 = lower limit on the altitude angle [radians]. Default = 0 for SpatialDist.type = surface; $-\pi/2$ otherwise.
- altitude1 = upper limit on the altitude angle [radians]. Default = $\pi/2$
- azimuth0 = lower limit on the azimuth range [radians].
- azimuth1 = upper limit on the azimuth range [radians].

See the notes above on the definitions of altitude and azimuth.

3.7 PerturbVel

3.8 plasma_info

3.9 options

The options structure contains various settings and parameters that don't fit anywhere else.

1 packets = number of packets to run. Default = 30000. In general, I limit the number of packets per iteration to 30000. I will run tests to see how necessary this is. When running in streamline mode, you shouldn't run more than 1000 or so packets at a time.

2 endtime = total integration period [seconds]. This should be set for at least 3 times the expected e-folding lifetime to make sure a steady state is reached.

3 resolution = the model precision. Default = 10^{-6}.

4 motion = 1/0 to turn on/off satellite motion. Default = 1. This is mainly used for testing purposes. It has no effect for Mercury.

5 lifetime = Fixed, constant lifetime for the packets [seconds]. To have the model determine the lifetime based on calculated loss rates (using electron-impact, photo, and charge exchage processes), set lifetime = 0. Default = 0.

6 atom = Species to run.

7 at_once = 1/0 to eject all the packets at once/random times. Default = 0 in the normal model and 1 in the streamline mode.

8 fullsystem = 1/0 to turn on/off full system mode. If turned on, the positions of packets are tracked regardless of how far from the starting points they go. If off, then packets which go too far from the starting point are no longer tracked. If only modeling the exosphere, you may not care about the packets that escape. If modeling a neutral torus, you will want to model the full system. Default = 1 for Jupiter and Saturn and 0 for Mercury.

9 outeredge = Distance from the starting point to track packets [object radii]. This only has an effect if options.fullsystem=0. Default = 20.

10 trackloss = 1/0 to turn on/off loss tracking. If set, the output will include:

- The fraction of each packet that was ionized or dissociated.
- The fraction of each packet that stuck to the surface of each object in the system.
- The fraction of each packet that hit Saturn's rings.
- The fraction of each packet that escaped from the exosphere (if options.fullsystem=0)
- The spatial distribution of surface adsorption on each object in the system.

Turning this on will increase the runtime, although I have not tested how much. This feature has not been well tested and may still have some bugs. Default = 0.

11 datapath = path to the atomic data used to determine loss rates, emission rates, and g-values. Default = path on my computer.

12 modelpath = path to the model code. Default = path on my computer.

4 Outputs

NEED TO REWRITE THIS SECTION

The output structures are:

1 output= final values for each packet

2 deposition = maps of adsorption on the surface of each object in the system (only returned if options.trackloss=1).

The startloc structure contains:

- x,y,z = Initial x, y, and z positions for each packet in a reference frame fixed on the starting object [object radii]. Need to figure out exactly what this returns.
- vx, vy, vz = Initial x, y, and z components of the velocity relative to the starting object [Planetary radii/second]. This does not include the starting objects orbital motion or the planet's motion relative to the sun.
- frac = Initial content
- TravelTime = integration time between 0 and options.endtime [seconds]
- phi = initial orbital longitude for the starting moon [radians]. Geometry.phi gives the orbital longitude at the end of the simulation. This array gives the longitude loc.TravelTime seconds earlier.
- longitude = initial surface longitude [radians]
- latitude = initial surface latitude [radians]
- altitude = ejection angle relative to normal [radians]
- azimuth = azimuthal ejection angle [radians]

The loc structure contains:

- x, y, z = final x, y, and z coordinate for each packet [R_{plan}]
- vx, vy, vz = final x, y, and z components of the velocity [R_{plan}/s]
- frac = final fractional content
- fintime = travel time for each packet [seconds]
- totalsource = Total initial fractional content

If options.trackloss=1, then loc also contains:

- lossfrac = fraction lost to ionization and dissociation processes
- hitfrac = fraction which stuck to each object in the system.
- ringfrac = fraction which hit Saturn's rings
- leftfrac = escaping fraction (if options.fullsystem=0)

Comments:

1 All of the fields in the structure which require arrays are actually pointers. See Section **??** for notes on using pointers in IDL.

2 The sum of the final fractional content and all the loss fractions should equal the initial starting content of the packet:
*loc.frac + *loc.lossfrac + *loc.hitfrac + *loc.ringfrac + *loc.leftfrac = *startloc.frac

The deposition structure contains:

- map
- longitude
- latitude

5 Using the Model

5.1 Running the Model

The basic command for running a model is:
modeldriver, inputfile, outputfile, seed, modstream=modstream, dt=dt, minimize=minimize

- inputfile = name of the input file
- outputfile = name of the output file
- seed = seed to use in the random number generator. If not specified, one will be chosen randomly.
- modstream = 0, 1, or 2 depending on which mode to use. Normal mode is 0, streamline modes are 1 and 2. Default = 0
- dt = time step if running in streamline mode [seconds].
- minimize = 1/0 to compress/not compress the output structures. This can save a lot of disk space, but you also lose information.

There are three modes for running the model.

- Normal mode - runs a normal model simulation. Only the begining and final state of each packet is recorded. This is the default mode, or it can be set explicitly by setting modstream=0
- Streamline Mode A - set modstream=1. Do not use this mode if packets bounce when striking the surface.
- Streamline Mode B - set modstream=2. This method treats packet bouncing correctly, but it is slower than method A.

To set up many models to run, use:
run_model, its, filelist=filelist, minimize=minimize, nodelete=nodelete, cycle=cycle

- its = number of iterations for each model. Default = 100
- filelist = name of a file containing the list of model input and output files (see below). Defualt = 'files.dat'
- minimize = 1/0 to compress/not compress the output files. Default = 0
- nodelete = 0/1 to delete/keep the intermediate files. Default = 0. In general, there is no reason to keep the intermediate files.
- cycle = 1/0 to cycle through the input files or do all the iterations of each input file before proceeding onto the next. Default = 0 (don't cycle)

The *filelist* file (files.dat) must be set up as follows:

```
inputfilepath outputfilepath
file0.input file0.output
file1.input file1.output
file2.input file2.output
```

The list of files can be as long as you want. I generally combine 100 iterations of 30,000 packets, which makes a large output file (~300 MB per file). To run more iterations, set up files.dat with the lines:

```
file0.input file0.0.input
file0.input file0.1.input
file0.input file0.2.input
```

and average the resulting images.

5.2 Looking at the Results

1 image = model_images(imtype, outputfile, strength, keyword=keyword)
This is the basic routine for makeing 2D model images of density, column density, intensity, or simulating spectra. Required inputs:

- imtype = 'column', 'density', 'intensity', 'spectrum'
- outputfile = name of the outputfile
- strength = source strength in units of 10^{26} atoms/sec. Usually I set this to 1 and multiply by source strength after the fact. The result is linear with respect to this parameter since all the emission is assumed to be optically thin.

Figure 1: Cartesion coordiate system used for model calculations.

2 result = line_of_sight(imtype, pos, dir, dr, outputfile, strength)
This determines the column density or intensity along the sightlines determined by direction dir from position pos.

3 den = density_track(x, y, z, dr, outputfile, strength)
Determines the density along the path x, y, z. This may not be working - needs to be verified.

6 Physics and Computational Section

In this section, I am going to try explaining everything that happens in a model run so that I have all the math, physics, and computational methods documented. This section will probably be pretty disorganized until I have everything down and can figure out how it should look. Much of this will be documentation of the code - I'll try to keep it current and reference the program version numbers.

6.1 Coordinate Systems

The model coordinate system
Model simulations use a cartesion coordinate system fixed on the center of the central planet (Figure 1). The x-axis points in the dusk direction (assumed opposite to the direction of planetary motion), the y-axis points in the anti-sunward direction, and the z-axis points north along the planetary rotation axis. The x-y plane is the rotational equatorial plane of the planet. In this system, radiation pressure always acts in the +y direction and satellite motion is always in the x-y plane.

Currently, the tilt of the planet relative to the exliptic is not included, although I can imagine that I may want to change this in the future.

Surface longitude and latitude
For the planets (Mercury, Jupiter, and Saturn) the surface latitude and longitudes are fixed with the subsolar point at (0°,0°). Longitude increases in the counter-clockwise sense if looking down over the north pole (increases in the direction of rotation) such that the dusk terminator is at 90°, midnight is 180°, and the dawn terminator is at 270°. Satellite longitude and latitude are fixed with respect to the sub-planet point and increase in the clockwise sense if looking down over the north pole: 90° is the leading (downstream) point, 180° is the anti-planet point, and 270° is the trailing (upstream) point. Latitude increases from the south pole (-90°) to the north pole (90°). No tilts or inclinations of the satellites relative to the ecliptic or planetary equatorial planes are included. (Note that all angles in the input files are in radians.) The time scales for neutrals are short enough that we can assume the orbital motion around the sun is negligible and the subsolar point is fixed on the surface of the planet (although not on satellites).

6.2 Common blocks

The common blocks are initialized in the procedure *model_common_blocks* (version 2.0, as of 14 June 2010).

1 **spectral**: radpres_const
Radpres_const is used for computing the radiation pressure acceleration (see Section **??**).

2 **constants**: SystemConsts, DipoleConsts
These are the constants related to the planet and its mangetic field. These constants are listed in Table **??**.

3 **ratecoefs**: coef_eimp, coef_chx, coef_photo
These contain the rate coefficients for the ionization and dissociation reactions used in the model run. These structures are defined in Section **??**

4 **plasma**: plasma, plasmahot
State of the thermal and hot plasma. This needs to be described explicitly somewhere.

6.3 modeldriver

Current as of 7/19/2010, modeldriver_3.0

1 Introductory Stuff
- 1.1 Load common blocks
- 1.2 Read in code versions
- 1.3 Determine if running packet streamlines – if so, will run a different program
- 1.4 Read in the *input* structure

2 Determine the number of packets that already exist
- 2.1 `overwrite EQ 0` ⇒ assume that no files exist.
- 2.2 `overwrite EQ 1`:
  - 2.2.1 determine number of existing specific files (*nfiles*) and generic files (*ngfiles*)
  - 2.2.2 `(nfiles EQ 0) and (ngfiles EQ 0)`: No packets exist
  - 2.2.3 `(nfiles GT 0) and (ngfiles EQ 0)`: All the existing files are specific files. Add up total number of existing packets
  - 2.2.4 `(nfiles EQ 0) and (ngfiles NE 0)`: Create specific files from the generic files and add up the total number of packets
  - 2.2.5 `(nfiles NE 0) and (ngfiles NE 0)`: Figure out which generic files have not been used, extract the specific files from them, and determine the total number of packets.

3 If the number of existing packets is greater than the number of requested packets, then everything is finished. Otherwise, set things up for more runs

4 Read in the system constants and dipole constants.

5 Determine the planet's r and dr/dt.

6 Find the path to the AtomicData. Possible options must be hard-wired in.

7 Set up the *loss_info* structure: run *lifetime_setup*

- 7.1 Find the default reactions
- 7.2 Determine the rate coefficients for the default reactions
- 7.3 If Mercury or Earth, then remove the electron impact and charge exchange reactions
- 7.4 If Jupiter or Saturn, load in the plasma

8 Set up the radiation pressure using *get_gvalue*:

- 8.1 Read in the g-values
- 8.2 The radiation pressure constant is given by:

$$c = \frac{h}{m\lambda} \sum g \left(\frac{0.352}{a} \right)^2 \tag{40}$$

  where the g-values have been normalized to 0.352 AU [**?**].
- 8.3 The function returns the radiation accelleration (cm s^{-2}) as function of velocity (cm s^{-1}).
- 8.4 Result divided by planetary radius to get velocity and accelleration in R_{plan}/s and R_{plan}/s^2

9 If `generic EQ 1` then set the inputs to generic spatial and velocity distributions. Otherwise, will be running outputs for the specific inputs.

10 Estimate how many interations of the new inputs will be needed. Set it up to do ten iterations of 100,000 packets to produce output files of 1,000,000 packets. Loop over the number of iterations that will be run

- 10.1 Determine the initial source distribution using *source_distribution*
  - 10.1.1 Spatial Distribution
    - If starting at a planet, then 0° longitude is sub-solar meridian and 90° is dawn meridian (right-handed coordinate system)
    - If starting at a satellite, then 0° longitude is sub-planet meridian and 90° is leading meridian (left-handed coordinate system)
  - 10.1.2 Speed Distribution
  - 10.1.3 Angular Distribution
- 10.2 Run the model: *driver* (see Section 6.4)

6.4 driver

Current version: *driver_3.0.pro*, 20 July 2010

1 Create the loc structure and set up a few other things

2 Set up bounce conditions

- Current options for *sticking_info.emitfn* are "Maxwellian" and "elastic scattering."

3 Begin main loop

3.1 Create structure (*loc0*) with only remaining packets

6.5 Choosing Points from Distribtuion Functions

[Section last edited 13 December 2010.]

An important element in the model is determining the initial state of each packet (location and velocity). This is done through several procedures in the SourceDistribtuion directory.

6.5.1 Choosing Random Numbers

The basic random number generator chooses psuedo-random numbers from between 0 and 1 with equal probability ($P(x) \sim 1$) for $0 \leq x \leq 1$. I use the random number generator from [**?**] (§7.1, p. 342), which is described as a "suspenders-and-belt, full-body-armor, never-any-doubt generator," and has a period $\approx 3.138 \times 10^{57}$. The routine is written in C++ with an IDL wrapper called *random_nr*. If the seed is not specified, one is chosen randomly using IDL's random deviate function *randumu*. *random_nr* has options in it to use faster methods with shorter periods (also from Numerical Recepies), but there is not much of a time savings and I don't want to have to worry about the random number generator. (As a note, I do not have any reason to suspect there is a problem with IDL's randomu procedure. I was simply learning how to link to C++ programs and liked the discussion of random deviates in Numerical Recepies.)

6.6 Choosing values from a 1D distribution

Points can easily be chosen from an arbitrary 1 dimensional probability distribution function using the transformation method. If the relative probability of a value x occurring is given by f(x), then the normalized cumulative distribution function is given by

$$F(x) = \frac{\int_{x_0}^{x} f(x)dx}{\int_{x_0}^{x_1} f(x)dx} \tag{41}$$

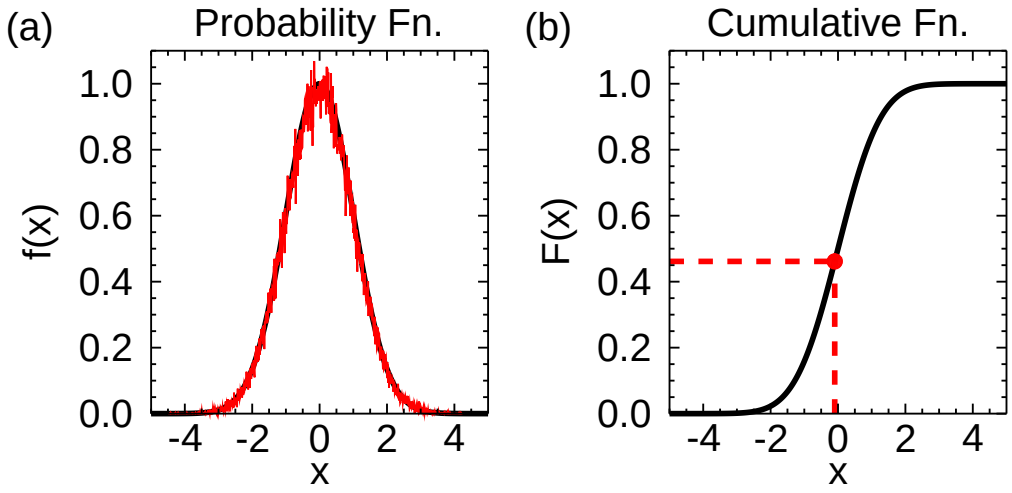

Figure 2: Transformation method to choose points from a probability distribution function. (a) Specified probability distribution function (black) and histogram of 10^5 points chosen from this distribution (red). (b) Cumulative distribution function (back). Red point shows a randomly drawn point from the distribution. Random F(x)=0.46 implies x=-0.10.

where x_0 and x_1 are the minimum and maximum values of x (Figure 2). The probability distribution f(x) does not need to be normalized in this case. The function $y = F(x)$ varies from 0 to 1. By choosing random y values, $x = F^{-1}(y)$, which can be computed numerically.

When using this method, the distribution needs to be well sampled, especially around discontinuites, because the values on the cumulative distribution are interpolated between the computed points.

Multivariate functions: P(x,y) = P(x)P(y)

Multivariate functions: P(x,y)

Spherical coordiates: $P(\phi, \theta)$

7 Appendix A: Directory Structure

Directory structure as of 19 January 2010. Directories are listed in blue. IDL program files are in black. Data files are in red. Data files ending in *.dat* are ASCII format. Data files ending in *.sav* are IDL save files. I've removed some files from this listing that don't do anything.

Notes: See Section **??** for a description of atomic data files and lists of the available reactions and references.

1 model_pro_2.0

- 1.1 AtomicData
  - 1.1.1 Lishawa_crosssections.pro
  - 1.1.2 extract_Berez_data.pro
  - 1.1.3 extract_Huebner_data.pro
  - 1.1.4 find_reactions_2.0.pro
  - 1.1.5 make_crosssec_struct_2.2.pro
  - 1.1.6 make_ratecoef_struct_2.1.pro
  - 1.1.7 make_reaction_list_2.1.pro
  - 1.1.8 rate_func_1.0.pro
  - 1.1.9 rate_integral_2.0.pro
  - 1.1.10 set_default_emission_2.0.pro
  - 1.1.11 set_default_reactions_2.0.pro
- 1.2 Data
  - 1.2.1 CAPSplasma
    - 1.2.1.1 SaturnPlasma.2008-04-16.sav
  - 1.2.2 VoyagerTorus
    - 1.2.2.1 VoyagerTorus.sav
    - 1.2.2.2 electrons.energetic.dat
    - 1.2.2.3 electrons.thermal.dat

8 Appendix B: Inputfile Summary

This appendix contains all the possibilities for each input. The lines can be copied and pasted into an input file as needed.

8.1 Geometry

- Mercury

```
geometry.planet = Mercury
geometry.taa = 0.
geometry.include = 1
```

- Jupiter

```
geometry.planet = Jupiter
geometry.startpoint = Io
geometry.CML = 0.
geometry.phi = 0. ;; Jupiter
geometry.phi = 4.71239        ;; Io
geometry.phi = 4.71239        ;; Europa
```

```
geometry.phi = 4.71239        ;; Ganymede
geometry.phi = 4.71239        ;; Callisto
geometry.include = 1          ;; Jupiter
geometry.include = 1          ;; Io
geometry.include = 0          ;; Europa
geometry.include = 0          ;; Ganymede
geometry.include = 0          ;; Callisto
```

- Saturn

```
geometry.planet = Saturn
geometry.startpoint = Enceladus
geometry.phi = 0.             ;; Saturn
geometry.phi = 4.71239        ;; Mimas
geometry.phi = 4.71239        ;; Enceladus
geometry.phi = 4.71239        ;; Tethys
geometry.phi = 4.71239        ;; Dione
geometry.phi = 4.71239        ;; Rhea
geometry.phi = 4.71239        ;; Titan
geometry.phi = 4.71239        ;; Hyperion
geometry.phi = 4.71239        ;; Iapetus
geometry.phi = 4.71239        ;; Phoebe
geometry.include = 1          ;; Saturn
geometry.include = 0          ;; Mimas
geometry.include = 1          ;; Enceladus
geometry.include = 0          ;; Tethys
geometry.include = 0          ;; Dione
geometry.include = 0          ;; Rhea
geometry.include = 0          ;; Titan
geometry.include = 0          ;; Hyperion
geometry.include = 0          ;; Iapetus
geometry.include = 0          ;; Phoebe
```

8.2 Sticking_Info

- For everything to stick to the surface:

```
sticking_info.StickCoef = 1
```

- For bouncing packets with thermal reemission:

```
sticking_info.StickCoef = 0
sticking_info.emitfn = Maxwellian
sticking_info.accom_factor = 1.
```

```
sticking_info.Tsurf = 0
```

- For bouncing packets to scatter elastically

```
sticking_info.StickCoef = 0
sticking_info.emitfn = elastic scattering
```

8.3 Forces

```
forces.gravity = 1
forces.radpres = 1
forces.lorentz = 0
```

8.4 SpatialDist

- For packets spread across the surface (or uniform exobase) between a specified longitude and latitude range:

```
SpatialDist.type = surface
SpatialDist.random_even = random
SpatialDist.use_map = 0
SpatialDist.exobase = 1.
SpatialDist.longitude0 = 0.
SpatialDist.longitude1 = 6.2831853
SpatialDist.latitude0 = -1.5707963
SpatialDist.latitude1 = 1.5707963
```

- For packets spread across the surface (or uniform exobase) according to a pre-specified probability distribution:

```
SpatialDist.type = surface
SpatialDist.random_even = random
SpatialDist.use_map = 1
SpatialDist.mapfile = mapfile.sav
SpatialDist.exobase = 1.
```

- For packets distributed in a torus around the central planet

```
SpatialDist.type = torus
SpatialDist.random_even = random
SpatialDist.radius0 = 4.
SpatialDist.radius1 = 1.
SpatialDist.radius2 = 1.
```

- For packets distributed in a spherically symmetric exosphere

```
SpatialDist.type = exosphere
SpatialDist.exotype = powerlaw
SpatialDist.b = -2
SpatialDist.rmax = 10.
SpatialDist.block_shadow = 0
```

- For packets distributed according to Vincent Dol's SO_2 exosphere distribution:

```
SpatialDist.type = SO2 exosphere
```

8.5 SpeedDist

- Gaussian speed distribution

```
SpeedDist.type = Gaussian
SpeedDist.vprob = 3.
SpeedDist.sigma = 1.
```

- Tri-gaussian speed distribution

```
SpeedDist.type = trigaussian
SpeedDist.vxprob = 3.
SpeedDist.vxsigma = 1.
SpeedDist.vyprob = 2.
SpeedDist.vysigma = 0.5
SpeedDist.vzprob = 1.
SpeedDist.vzsigma = 0.1
```

- Sputtering speed distribution

```
SpeedDist.type = sputtering
SpeedDist.U = 2.
SpeedDist.alpha = 3
SpeedDist.beta = 1.
```

- Maxwellian at constant temperature

```
SpeedDist.type = maxwellian
SpeedDist.temperature = 2000.
```

- Thermal distribution at local surface temperature

```
SpeedDist.type = thermal
```

- Equal probability for $v_{prob} \pm \Delta v$

```
SpeedDist.type = flat
SpeedDist.vprob = 3
SpeedDist.delv = 1.
```

- Start packets in stable, circular orbits about the central planet

```
SpeedDist.type = circular orbits
```

- Use Vincent Dol's SO_2 exosphere velocity distribution

```
SpeedDist.type = dolsfunction
SpeedDist.dols0 = 1.
SpeedDist.dols1 = 1.
```

8.6 AngularDist

- The speed distribution selected doesn't need an angular distribution to be specified.

```
AngularDist.type = none
```

- Send packets out radially from the central obeject

```
AngularDist.type = radial
```

- Send packets out isotropically between specified altitude and azimuth ranges

```
AngularDist.type = isotropic
AngularDist.altitude0 = 0.
AngularDist.altitude1 = 1.5707963
AngularDist.azimuth0 = 0.
AngularDist.azimuth1 = 6.2831853
```

- Send packets out proportional to $\cos^n \theta$, where θ is the altitude.

```
AngularDist.type = costheta
AngularDist.n = 1.
AngularDist.altitude0 = 0.
AngularDist.altitude1 = 1.5707963
AngularDist.azimuth0 = 0.
AngularDist.azimuth1 = 6.2831853
```

8.7 PerturbVel

- Don't add a perturbation onto the initial velocity distribution.

```
PerturbVel.type = none
```

- Add a Gaussian perturbation onto the initial velocity distribution.

```
PerturbVel.type = gaussian
PerturbVel.vprob = 1.
PerturbVel.sigma = 0.2
```

- Add a tri-gaussian perturbation onto the initial velocity distribution

```
PerturbVel.type = trigaussian
PerturbVel.vxprob = 1.
PerturbVel.vxsigma = 0.2
PerturbVel.vyprob = 1.
PerturbVel.vysigma = 0.2
PerturbVel.vzprob = 1.
PerturbVel.vzsigma = 0.2
```

- Use a charge exchage type perturbation

```
PerturbVel.type = charge exchange
PerturbVel.flowvel = 10.
```

8.8 plasma_info

- Mercury – *not used*
- Jupiter

```
plasma_info.eps = 0.246
plasma_info.thermal = 1
plasma_info.energetic = 1
```

- Saturn

```
plasma_info.elecdenmod = 1
plasma_info.electempmod = 1
```

8.9 options

```
options.packets = 30000
options.endtime = 36000
options.resolution = 1e-6
options.at_once = 0
options.atom = Na
options.lifetime = 0
options.fullsystem = 0
options.outeredge = 20.
options.motion = 1
options.trackloss = 0
options.datapath = $HOME/Data/AtomicData
options.modelpath = $HOME/Work/NeutralModel/modelpro_2.0
```