

ExoVista User's Guide

Chris Stark, NASA GSFC

1. Summary

ExoVista is a hybrid IDL/C software package that generates synthetic exoplanetary systems. ExoVista models exoplanet atmospheres in reflected light, stellar spectra using Kurucz stellar atmosphere models, and debris disks in scattered light using realistic spatial distributions and optical properties. Planets can be drawn from measured/extrapolated *Kepler* occurrence rates and are checked for basic stability criteria; debris disks are dynamically quasi-self-consistent with the underlying planetary system. All bodies are integrated with a Bulirsch-Stoer integrator to determine their barycentric velocities, positions, and orbits. The output product is a multi-extension fits file that contains all of the information necessary to generate a spectral data cube of the system for direct imaging simulations of coronagraphs/starshades, as well as position/velocity data for simulation of RV, astrometric, and transit data sets.

2. Philosophy & Numerical Approach

ExoVista was designed with two primary numerical goals. First, exoVista was designed for speed, to rapidly generate planetary systems for a large number of stars/scenarios. To enable this, only simple checks are performed for dynamic stability of the planetary systems and analytic models are used for the debris disks. Second, exoVista's output file size was minimized, such that all required information on the planetary system is contained in $< \sim 10$ MB. To minimize file size, exoVista does not save a simple spectral image cube. Instead, the output is a list of each point source's position, velocity, orbit, and contrast/flux, combined with a contrast data cube of the debris disk (which has a smooth wavelength dependence and can be saved at lower spectral resolution). As a consequence of saving disk contrast instead of disk flux, exoVista cannot include thermal emission from the disk. Also, some "massaging" of the planetary scene is required to convert it to an image (if desired). We have provided routines in IDL (*load_scene.pro*) and Python (*load_scene.py*) that load the output .fits file to aid in converting it to a spectral image cube.

3. Use Cases

Users of exoVista will likely fall into one of three categories:

1. Those that want to use existing simulations of planetary systems to simulate direct imaging instruments
2. Those that want to use existing simulations of planetary systems for RV, astrometric, or transit simulations
3. Those that want to use the exoVista code to generate their own planetary systems

For Use Case #1, the user will download existing .fits planetary scene files, and then use one of the "load_scene" tools (or create their own). The "load_scene" tool loads .fits planetary scene files and places it into the desired format to simulate the response of a coronagraph or starshade.

For Use Case #2, the user will download existing .fits planetary scene files and then manually interact with the .fits files to extract the desired information. Examples of this are provided later.

For Use Case #3, the user must become familiar with the details of how exoVista works, all of its modules, dependencies, and have adequate computational facilities.

Examples of these use cases are provided at the end of this manual. However, regardless of what you intend to do, you should have an understanding of exoVista's modules, the assumptions made, methods used, and format of the output files.

4. Installation

4.1 Use Case #1 & Use Case #2

An IDL license is not required for these use cases if you intend to interact with existing .fits files using another coding language. We still recommend downloading the exoVista code, which provides example load_scene routines that the user can run or use as a guide.

To obtain the exoVista code, download the most recent exovista.zip package. Unzip it and place the extracted files in the desired directory on your machine. Then obtain the desired package of .fits files and place them in the desired location. (Since you will not be running the main modules of the exoVista code, you do not need to have a C compiler present on your system as described in the next section.)

4.2 Use Case #3

To run the main modules of exoVista, you must have an IDL license and a C compiler (Apple users could install the XCode package to obtain a C compiler).

To install exoVista, download the exovista.zip package containing the software. Unzip it and place the extracted files in the desired directory on your local machine.

Open a terminal window, navigate to the directory containing the exoVista code, and start IDL by typing "IDL." Compile the Bulirsch-Stoer integrator by executing the following command:

```
IDL> compilec, 'nbody'
```

This should call your C compiler successfully and generate the file 'nbody.so' on your machine in the same location as the 'nbody.c' file. You are now ready to run all exoVista modules.

5. Modules

5.1 load_stars.pro

Syntax: *load_stars, s*

The first step of running exoVista is to define your star(s). You can define your own star(s) or use *load_stars.pro* to load the ~8k stars from the HabEx/LUVOIR master target list (these are stars w/in 50 pc and predominantly brighter than 8th mag). If you are interested in defining your own star(s) instead of using *load_stars*, take a look at the *generate_solarsystem.pro* routine, which provides an example of how to define your own star structure.

load_stars will return an array of star structures to the desired variable (here we adopt the variable *s*). Each entry in *s* (e.g., *s[0]*, *s[1]*, etc.) is a structure containing a variety of stellar parameters. The following is an example of how to call *load_stars*, along with a list of the star structure tags:

```
IDL> load_stars,s
Loading target list...done.
IDL> print,tag_names(s)
STARID HIP TYC RA DEC UMAG BMAG VMAG RMAG IMAG JMAG HMAG KMAG M_V DIST TYPE LSTAR TEFF ANGDIAM MASS LOGG RSTAR WDS_SEP WDS_DMAG
IDL> █
```

Not all of the above tag names are necessary to run exoVista. See *generate_solarsystem.pro* for an example on which ones need to be defined.

The HabEx/LUVOIR master target list was generated by combining the Hipparcos and Gaia TGAS/DR2 target lists within 50 pc. All Hipparcos distances were updated with Gaia distances. The Washington Double Star catalog was cross-referenced to obtain simple binary parameters (separation and dmag of nearest companion); no other binary catalogs were referenced. Spectral type was taken from the original Hipparcos release and updated with SIMBAD. Additional photometry was pulled from SIMBAD. All luminosity class III and IV stars were removed. Stellar angular diameters are estimated using the B-V analytic fits from Boyajian et al. (2014). Physical stellar diameters are based off the angular diameter and distance. An updated table based off of Pecaut & Mamajek (2013) is used to estimate stellar mass and effective temperature by interpolating B-V. Log(g) is estimated from this mass and the estimated stellar radius. Luminosity is estimated from the absolute V band magnitude and a color correction using Eq 9 of Torres (2010) and the color correction factors in footnote 10 of Pecaut & Mamajek (2013). **This target list is not intended to be accurate on a star-by-star basis.** It is ended to roughly represent the population of nearby main sequence and sub-giant stars. This target list could be substantially improved in accuracy, but a thorough analysis of every star in this list is beyond the scope of this project.

5.2 generate_planets.pro

Syntax: *generate_planets, s*

After creating your stars, you may call *generate_planets* to distribute planets around each star. *generate_planets.pro* modifies the array of star structures to include:

- *i*: the inclination (in degrees) of the system midplane
- *PA*: the position angle (in degrees) of the system midplane
- *planet*: a 30-element array of planet structures (each star can accommodate up to 30 planets) containing the planets' data.

Below is an example of how to call *generate_planets.pro* along with a list of the planet structure tags.

```
IDL> generate_planets,s
Generating planets...
Retrieving occurrence rates in non-standard space of (mass, semi-major axis)...
Loading occurrence rates: occurrence_rates/NominalOcc_Mass.csv
87504.3 planets expected from occurrence rates
87678.0 planets expected from Monte Carlo draw of occurrence rates
87678 planets generated after imposing stability limits.
...done.
% Program caused arithmetic error: Floating divide by 0
IDL> print,tag_names(s[0].planet)
M R ALBEDO_FILE A E I LONGNODE ARGPERI MEANANOM
IDL> print,s[0].planet[7]
{ 0.512824 0.836646 ./geometric_albedo_files/Mars.txt 9.09240 0.00000 4.42195 162.836 94.5883 157.303}
IDL>
```

The planet structure consists of the planet mass (M), radius (R), the filename containing the planet's reflected light geometric albedo, and orbital elements (in AU and degrees). Planets are sorted by semi-major axis a . **All orbital elements are relative to the system midplane orientation.**

Planets are drawn from the mass-semi-major axis occurrence rate maps of Dulz et al. (2020). By default the occurrence rate maps are treated as a probability map. For all n stars a Monte Carlo draw is performed on every bin in (M, a) space of the map to find the total number of planets to distribute in each bin. As a result, when running exoVista the number of planets generated may vary from run to run, and be greater or less than the expectation value.

After determining the total number of planets to generate for each (M, a) bin, the code randomly assigns those planets to stars. Planet mass and semi-major axis is randomly spaced logarithmically within each bin. Planet mass is used to determine radius using the relationship of Chen & Kipping (2017). Eccentricities are zero by default. Planet inclinations are uniformly distributed between 0 and 5 degrees by default, relative to the system midplane. All other orbital angles are randomly distributed between 0 and 360 degrees.

After adding planets, the code checks all planet pairs to determine if they are dynamically stable. To do so, exoVista calculates the mutual Hill sphere and requires $\Delta > 6$. All planets with $\Delta < 6$ are deleted. The code then iteratively adds more planets until the desired number per bin are generated. If any bin has not achieved its desired number of planets in 50 iterations, the creation of planets ends.

We note that the occurrence rates of Dulz et al. (2020) adopted $\Delta > 9$, as motivated by the criteria for Gyr of stability. Adopting the same criteria here greatly slows the planet distribution process. Because we are generating systems that are "plausible," and not necessarily stable for many Gyr, we relaxed this criterion to speed up the code.

Planet albedo files are then randomly assigned based on which albedo files are applicable to which (M, a) bin. Simple rules assigning each albedo file are defined in *assign_albedo_file.pro*. Each albedo file can be assigned a probability from 1-100 that determines the relative likelihood of drawing that planet, but this must be compared to the total probabilities of all other planets assigned to the same phase space. Note that exoEarth candidates (EECs) are treated separately in their own phase space to more easily control their distribution. Currently all EECs are assigned

some form of an Earth-like atmosphere (Archean, Hazy Archean, Proterozoic High-O₂, Proterozoic Low-O₂, and Modern Earth) based on the relative timescales of these phase of Earth's history. See `assign_albedo_file.pro` for details.

Note that we apply the occurrence rates in flux-normalized semi-major axis. I.e., we interpret the occurrence rates of Dulz et al. (2020) to have a constant η_{Earth} regardless of spectral type. Planet semi-major axis is drawn from the occurrence rates, and then applied to an individual star by multiplying by the square root of the bolometric luminosity.

Note that the planet generation process does not currently attempt to reproduce the empirically measured multiplicity—the tendency of planetary systems to host more than the average number of planets per star.

5.3 `generate_disks.pro`

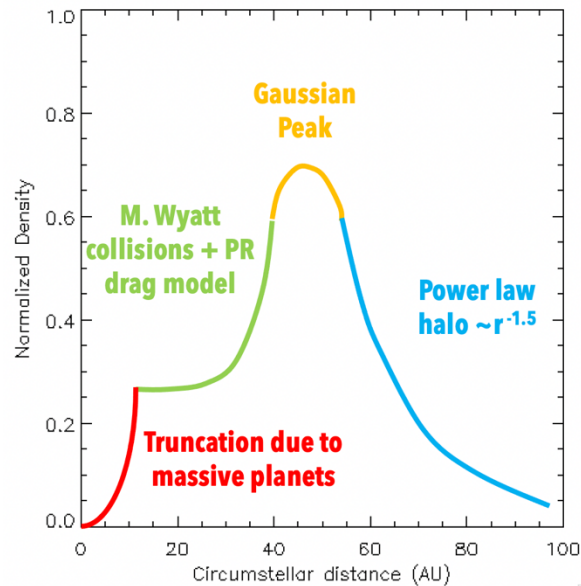
Syntax: `generate_disks,s`

After distributing planets to each star that are quasi-dynamically stable, you can call `generate_disks.pro` to distribute debris disks to each star. `generate_disks` modifies the input data structure to contain an additional tag named `disk`. `disk` is up to a 3-element array of disk structures, containing “warm,” “cool,” and “cold” debris disks. Each entry in `disk` contains basic information that defines the disk's orientation, density, geometry, and scattering properties. Below is an example call to `generate_disks` and the tag names associated with the disk data structure.

```
IDL> generate_disks,s
Generating 2 disk components per star...done.
% Program caused arithmetic error: Floating illegal operand
IDL> print,tag_names(s[0].disk)
LONGNODE I NZODIS R DROR RINNER ETA HOR G W
IDL> print,s[0].disk[1]
{      0.00000      0.00000      18.0921      49.7390      0.0528117      0.0100000      0.148704      0.170033      0.841447      0.333514      0.133600
  0.411592      0.103476      0.484932
}
IDL>
```

Here i and *longnode* are the inclination and longitude of the ascending node for the disk relative to the system midplane; all disks are currently assigned the same orientation as the system midplane. *nzodis* is the density of the debris disk, as drawn from the best-fit LBTI distribution. *g* is a 3-element array containing the scattering asymmetry parameters for a linear combination of 3 Henyey-Greenstein scattering phase functions; their relative weights are contained in *w*. The values of *g* and their weights are randomly drawn from values nearby the best fits to Saturn's G ring from Hedman & Stark (2015).

By default exoVista distributes 2 disk components. ExoVista determines plausible locations for the parent body rings based on the underlying planetary system. To do this, exoVista determines all semi-major axes that are not within 3 Hill radii of a planet. For the inner warm component, exoVista finds the most stable region (largest width of stable semi-major axes in log space) between 0.5 and 5 AU. For the cool component, the process is repeated from 5-50 AU. If the user desires a third component, it is placed between 50-500 AU.



Each component is described by a piece-wise combination of analytic models as shown in the Figure above. This includes a Gaussian parent body ring at distance R and of width dR , within which all dust is assumed to follow a Dohnanyi size distribution. Outside of $R+dR$ the disk is modeled as a power law halo of dust, and inward of $R-dR$ the disk is modeled as inward-migrating, colliding dust. A Dohnanyi size distribution is adopted for the Gaussian ring. Although the outer halo component should have a non-Dohnanyi size distribution, the expected size distribution is not well-described analytically and exoVista currently assumes it is Dohnanyi as well. The interior component is modeled with the analytic collisional model of M. Wyatt, and the rate of collision is calculated quasi-self-consistently with the exozodi density. The collision rate is calculated independently for each grain size, such that the inward-migrating component can have a significantly non-Dohnanyi size distribution for massive disks, and thus exhibit a radial color gradient. Finally, exoVista models dynamically-ejected dust by applying an $(r/r_{\text{truncate}})^3$ weighting to dust, where $r_{\text{truncate}} = 1.1 * a * (1+e)$ for the outer-most planet interior to the belt that is > 100 Earth masses. The $(r/r_{\text{truncate}})^3$ behavior is motivated from the modeled optical depth reduction of inward-migrating Kuiper Belt dust due to Jupiter and Saturn (see Kuchner & Stark 2010).

Note that exoVista does not model mean motion resonant ring structures created by planets. All disk models are assumed to be azimuthally symmetric and centered on the star.

5.4 generate_scene.pro

Syntax: *generate_scene,s*

Once the stars have been loaded, the planets have been distributed, and the disks have been placed, the user may generate the .fits scene files. This is the most numerically demanding portion of the exoVista package, with most time spent generating the disk image.

generate_scene defines a wavelength array for the star and planets, with the default being 0.3-1.0 microns with $R=300$. The user can change this, though **exoVista does not handle thermal emission from the disk or planets**. *generate_scene* also defines the wavelength array for the disk, which defaults to the same range as the star and planets, but with $R=10$. The lower spectral resolution for the disk is allowed because the disk's spectral features are broad at visible wavelengths. The lower spectral resolution allows us to save substantial disk space for the output products. *generate_scene* then loads the geometric albedo files for the planets and interpolates to the desired spectral resolution.

generate_scene also defines the dust grain size resolution (s/ds) for the debris disk images. By default $s/ds = 5$, which has been thoroughly tested to produce adequate results. We note that the Mie theory calculations required to find Q_{abs} and Q_{sca} as a function of grain size and wavelength have all been done in advance, so if the user desires to change the grain size resolution, those calculations will have to be redone. See details of this in the appendix. The range of grain sizes extends from the blowout size to 100x the longest wavelength considered. **Currently all stars are assumed to have a blowout size of 0.5 microns.**

generate_scene then loops through all stars. For each star, the code first retrieves the best Kurucz stellar model based on the stellar effective temperature, luminosity, and $\log(g)$. The stellar spectral model is normalized to reproduce the star's observed V band flux, not normalized to the bolometric luminosity or radius, which are approximate quantities derived from analytic fits to B-V for main sequence stars.

Second, the code distributes dust grains based on the geometry and parameters of each disk component and generates each disk component image. The disk images, calculated at lower spectral resolution, are divided by the stellar flux at those wavelengths, resulting in a disk contrast cube. While the flux cube would have the star's spectral features imprinted on it, and thus need to be calculated at higher spectral resolution, the contrast cube does not, allowing for lower spectral resolution.

Next, *generate_scene* evolves all planets and the star using a Bulirsch-Stoer integrator. By default, this only occurs for one time, $t=0$. If the user specifies the keyword *timemax* in units of years, the code integrates at high time resolution steps and saves the results every 10 days for *timemax* years. Because the disk is assumed to be azimuthally symmetric, no integration is needed for the disk particles.

Finally, *generate_scene* produces the output .fits scene file. A detailed description of the output data products are included in the section below.

6 Output Data Products

For each star, the *generate_scene* module of exoVista produces a single .fits file. This file has multiple extensions. The data contained in each extension is as follows:

Extension #	Data description
0	Vector of wavelength values for star and planets
1	Vector of wavelength values for debris disk
2	3D debris disk contrast cube (x, y, lambda) + noise map
3	2D star data array (time, position & orbit & spectrum)
4...N_EXT	2D planet data array (time, position & orbit & spectrum)

Extension 0: Wavelength values for star and planets

Description: a 1D vector containing the wavelengths (in microns) at which stellar flux and planet contrast were calculated

Key header parameters:

NAXIS1: length of wavelength vector (# of wavelengths)

N_EXT: maximum extension number (planet data in extensions 4 - N_EXT)

SPECRES: spectral resolution of wavelength vector

LAMMIN: minimum wavelength

LAMMAX: maximum wavelength

Extension 1: Wavelength values for debris disk contrast cube

Description: a 1D vector containing the wavelengths (in microns) at which disk contrast was calculated

Key header parameters:

NAXIS1: length of wavelength vector (# of wavelengths)

SPECRES: spectral resolution of wavelength vector

LAMMIN: minimum wavelength

LAMMAX: maximum wavelength

Extension 2: Debris disk contrast cube

Description: a 3D cube (xpix x ypix x wavelength) of disk flux divided by star flux. To convert this back into a disk flux, interpolate the cube to the desired wavelength, then multiply by the stellar flux at those wavelengths. Note: The number of entries in the last dimension is equal to the number of wavelengths + 1. The "+ 1" is due to the fact that the last entry is not a contrast map, but a 2D map estimating the numerical noise in the contrast calculations.

Key header parameters:

NAXIS1: # of pixels in x dimension

NAXIS2: # of pixels in y dimension

NAXIS3: # of wavelengths + 1

SPECRES: spectral resolution of wavelength vector

PXSCLMAS: pixel scale in mas

LNGND-N: longitude of the ascending node of the Nth disk component (degrees)

I-N: inclination of the Nth disk component relative to system midplane (degrees)

NZODIS-N: density of zodis of the Nth disk component

R-N: circumstellar distance of the peak density of the Nth disk component (AU)

DROR-N: value of the normalized Gaussian peak width of the Nth disk component

RINNER-N: value of the inner truncation radius of the Nth disk component (AU)

ETA-N: ratio of PR drag time to collision time for the blowout size for the Nth disk component

HOR-N: normalized scale height for the Nth disk component

G0-N - G2-N: 3 values of scattering asymmetry parameter for the Nth disk component

W0-N - W2-N: 3 weights for each HG scattering phase function for the Nth disk component

MINSIZE: minimum grain size considered

MAXSIZE: maximum grain size considered

Extension 3: star data

Description: a 2D array (time x data value) containing the time, position, orbit, and spectrum of the star for all time values. The data structure is organized as follows:

data[i,j]: ith time value, jth data value
data[i,0]: value of time (years)
data[i,1]: x coordinate of star (in pixels) at ith time
data[i,2]: y coordinate of star (in pixels) at ith time
data[i,3:8]: heliocentric orbital elements at ith time (N/A for star, set to zero)
data[i,9:14]: barycentric x, y, z positions (in AU) and barycentric vx, vy, vz velocities (in AU/yr) at ith time
data[i,15:14+nlambda]: flux of star (in Jy) at ith time

Key header parameters:

NAXIS1: # of time values
NAXIS2: # of data values for each time value
PA: position angle of system midplane (degrees)
I: inclination of system midplane (degrees)
STARID: an internal catalog ID # for the star
RA: right ascension of star (decimal degrees)
DEC: declination of star (decimal degrees)
XMAG: stellar empirical magnitude for the X band
M_V: absolute V band magnitude of star
DIST: distance to star (pc)
TYPE: spectral type of star
LSTAR: bolometric stellar luminosity (solar luminosities)
TEFF: stellar effective temperature (K)
ANGDIAM: angular diameter of star (mas)
MASS: stellar mass (solar masses)
LOGG: log(stellar gravity)
RSTAR: stellar radius (solar radii)
WDS_SEP: most recent separation of companion in WDS catalog, if it exists (arcsec)
WDS_DMAG: delta mag of companion in WDS catalog, if it exists
PMRA: proper motion in RA (mas/yr)
PMDEC: proper motion in DEC (mas/yr)
PXSCLMAS: pixel scale (mas)

Extension 4 - N_EXT: planet data

Description: a 2D array (time x data value) containing the time, position, orbit, and **contrast** spectrum of each planet for all time values. The data structure is organized as follows:

data[i,j]: ith time value, jth data value
data[i,0]: value of time (years)
data[i,1]: x coordinate of planet (in pixels) at ith time
data[i,2]: y coordinate of planet (in pixels) at ith time
data[i,3:8]: heliocentric orbital elements at ith time; semi-major axis (AU), eccentricity, inclination (degrees), longitude of ascending node (degrees), argument of pericenter (degrees), mean anomaly (degrees)
data[i,9:14]: barycentric x, y, z positions (in AU) and barycentric vx, vy, vz velocities (in AU/yr) at ith time
data[i,15:14+nlambda]: **contrast** of planet at ith time

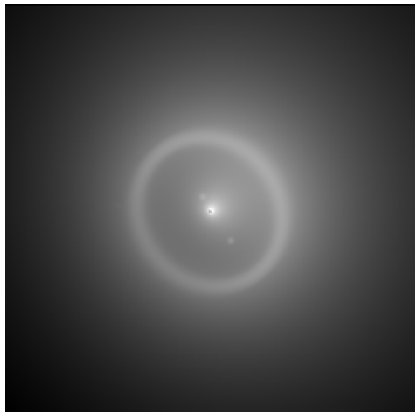
Key header parameters

NAXIS1: # of time values
NAXIS2: # of data values for each time value
M: planet mass (Earth masses)
R: planet radius (Earth radii)
ALBEDO_F: geometric albedo filename
PXSCLMAS: pixel scale (mas)

7 Example Uses of Outputs

7.1 Generating an astrophysical scene for high contrast imaging

We have included an example of a routine, *example_image.pro*, that produces an image cube for modeling high contrast imaging surveys. This routine calls *load_scene* to retrieve the necessary output, then convolves the planet fluxes with an Airy pattern PSF, then sums up all components (except for the star). The resulting image is shown below.



7.2 RV analysis

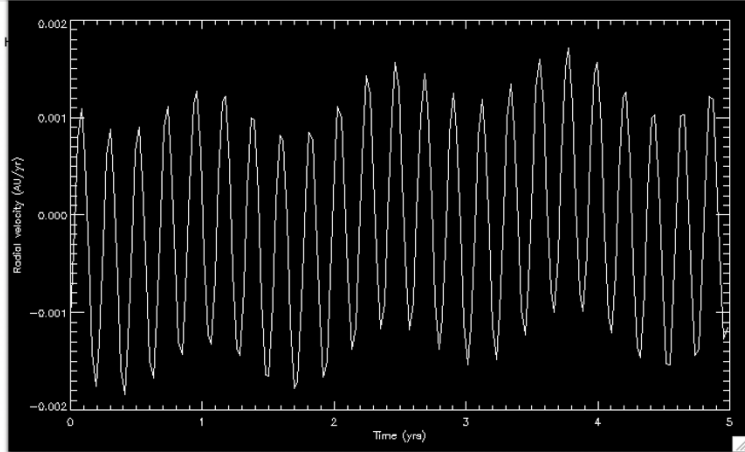
ExoVista calculates the barycentric velocity of the star as a function of time, so one can plot the simulated radial velocities of every star. The star's data is located in the 3rd extension of the .fits file. Here is an example in which the stellar data is retrieved for a single system ("star #0") and we plot the stellar radial velocity, which clearly shows the presence of at least three planets:

```

cstark@gs66-jarvis exovista % idl
IDL Version 8.5.1, Mac OS X (darwin x86_64 m64).
(c) 2015, Exelis Visual Information Solutions, Inc., a subsidiary of
Installation number: 660-1249-13.
Licensed for use by: NASA - GSFC

IDL> d = readfits('./output/0.fits',h,ext=3)
IDL> t = d[*,0]
IDL> vz = d[*,14]
IDL> plot,t,vz,xtitle='Time (yrs)',ytitle='Radial velocity (AU/yr)'
% Unsupported X Windows visual (class: TrueColor, depth: 8).
  Substituting default (class: TrueColor, Depth: 24).
IDL>

```



7.3 Transit & TTV analysis

The barycentric coordinates of all bodies are recorded as a function of time, as well as the heliocentric pixel coordinates. Below is an example in which we interpolate the heliocentric coordinates of a few planets in an edge-on system and compare it to the stellar radius. As you can see, one of the planets (green line) has either a transit or eclipse event (the barycentric z coordinate will differentiate between this).

```

pro makeplots

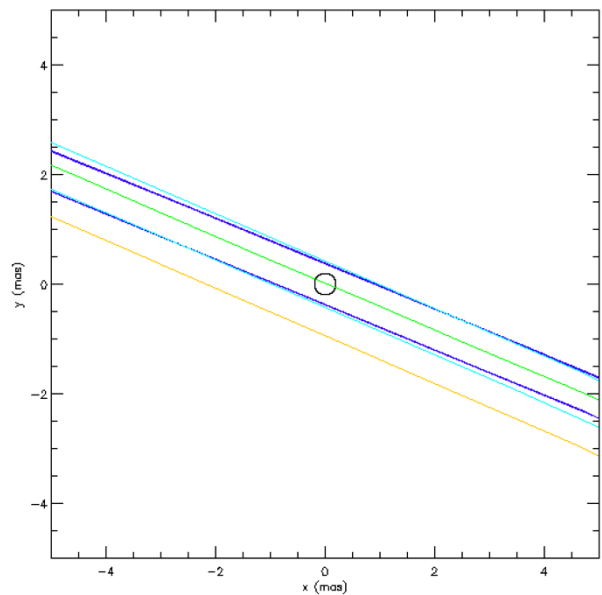
fitsfile = './output/0.fits'
h = headfits(fitsfile)
nplanets = sxpar(h,'N_EXT')-4

;Stellar information
d = readfits(fitsfile,h,ext=3)
a = sxpar(h,'ANGDIAM') ;mas
pixscale = sxpar(h,'PXSCALE') ;mas
xstarpix = d[*,1] ;x pixel coordinate of star vs time
ystarpix = d[*,2] ;y pixel coordinate of star vs time

;Plot stellar diameter
phi = findgen(361)*!pi/180.
r = a + fltarr(361)
window,0,xsize=600,ysize=600
loadct,39
plot,r*cos(phi),r*sin(phi),xtitle='x (mas)',ytitle='y (mas)', $
  /xs,/ys,xrange=[-5,5],yrange=[-5,5],background=!white,color=0, $
  position=[0.15,0.15,0.95,0.95],/norm

;Plot x and y coordinates of planets
for i=0,nplanets-1 do begin
  color = (i+1)*50
  d = readfits(fitsfile,h,ext=4+i)
  xmas = (d[*,1]-xstarpix)*pixscale
  ymas = (d[*,2]-ystarpix)*pixscale
  oplot,xmas,ymas,color=color
endfor
end

```



For planets whose orbital periods were adequately resolved by the record time step interval, one can also interpolate the heliocentric positions to a higher resolution time grid and use the results to perform a TTV analysis, as shown below.

```

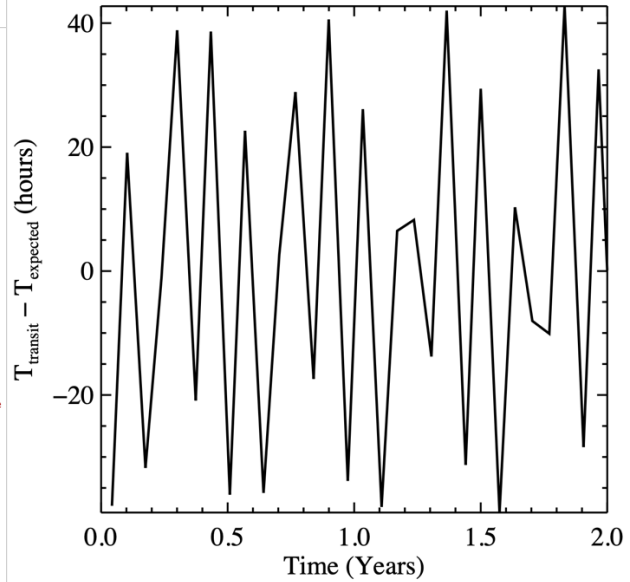
;Read the data
iplanet.extension = 5
d = readfits(fitsfile,h,ext=iplanet_extension)
pixscalemas = sgar(h,'PIXSCALEMAS')
t = reform(d[0,*])
x = reform(d[1,*]*npix/2)*pixscalemas)
y = reform(d[2,*]*npix/2)*pixscalemas)
r = sqrt(x**2+y**2)
z = reform(d[11,*])

;Interpolate to very fine time grid
t_hd = findgen(10000*long(n_elements(t)))
t_hd *= max(t)/max(t_hd)
r_hd = interpol(r,t,t_hd,/quadratic)
z_hd = interpol(z,t,t_hd,/quadratic)

;Find transit ingress times
valid = t_hd*0
j = where(z_hd gt 0 and abs(x_hd) lt angdiam)
if j[0] ne -1 then valid[j] = 1
undefine,ttransit
for j=0,n_elements(t_hd)-1 do begin
  if valid[j] eq 1 then begin
    if n_elements(ttransit) eq 0 then ttransit = t_hd[j] else ttransit = [ttransit,t_hd[j]]
  endif
  while valid[j] eq 1 do j++
endfor
periods = ttransit[1:n_elements(ttransit)-1]-ttransit[0:n_elements(ttransit)-2]
avgperiod = mean(periods)
tdiff = ttransit*0
for j=0,n_elements(ttransit)-1 do tdiff[j] = (j+1)*avgperiod-ttransit[j] ;subtract off expected transit time
tdiff -= mean(tdiff) ;make zero-mean
tdiff *= 365.25*24 ;convert to hours

;Plot the TTVs
set_plot,'ps'
savefile='TTV_plot.eps'
lp.font=0
lp.charsize=1.0
lp.thick=4 & lx.thick=3 & ly.thick=3
lp.multi=[0,1,1]
device,/times,/isolant,/encapsul,bits_per_pixel=8,/color,/cmk,$
  filenamesavefile,/inches,xsize=4,ysize=4
plot,ttransit,tdiff,xtitle='Time (Years)',ytitle='TIDtransitIN - TIDexpectedIN (hours)',$
  /xs,/ys,color=black,position=[0.15,0.15,0.95,0.95],/norm,xrange=[0,2]
device,/close

```



Appendix

A1. Disk contrast calculations

Disk contrast is calculated on a pixel-by-pixel basis by integrated along the line of sight. The z-resolution is set by a minimum number of pixels in the z direction and a maximum z value, set equal to 2 times the largest extent of the disk in the z direction within the image. The code iteratively resolves the z dimension with greater resolution until the integrated contrast changes by less than the desired tolerance. The code then increases the sub-pixel (x,y) resolution and checks that the tolerance is still maintained. The tolerance is set to a maximum of 0.05 for disks fainter than 1 zodi, which is equivalent to 1/3rd the flux of a dmag=26.5 point source using a 4 m telescope. The tolerance is inversely proportional to the zodi level down to a minimum of 0.0005—in the case of >100 zodis, at which point is assumed unlikely that a dmag=26.5 point source will be imaged for that system.

The disk contrast is calculated for each grain size using a combination of Mie theory and analytic approximations. ExoVista uses Mie theory-calculated scattering efficiencies, Q_{sca} . **To improve run time, exoVista does not use a Mie-theory scattering phase function (SPF), and instead opts for a scattering phase function composed of a linear combination of three Henyey-Greenstein SPFs.** This choice

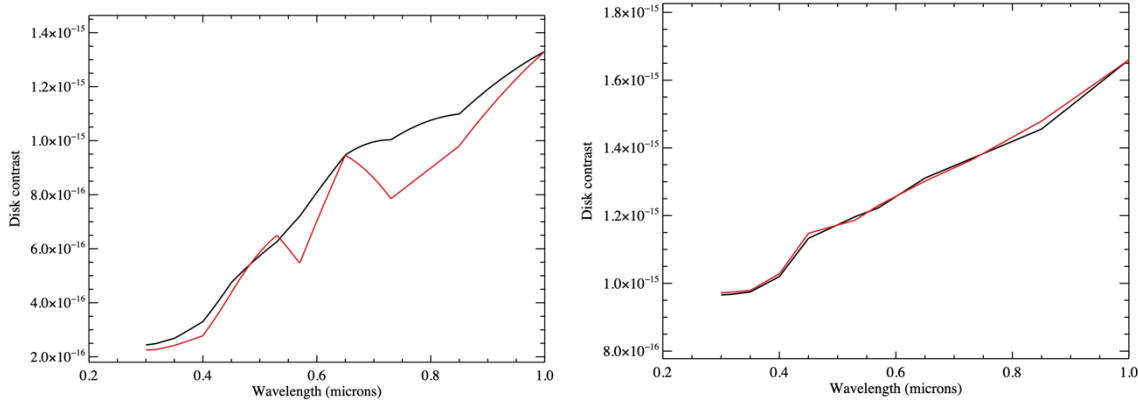
does impact the appearance of the disk, so we spend some time explaining our rationale.

Ideally we'd have a detailed description of the oddly-shaped porous aggregates of real debris disks. In practice this is a nearly impossible task, as each disk can have grains covering a range of compositions and shapes. We know that the Q_{abs} values calculated by Mie theory can be used to fit the thermal SEDs of debris disks and suggest plausible compositions dominated by astronomical silicates, water ice, and vacuum, as well as plausible size distributions (near Dohnanyi). These calculations also predict important disk color variations at visible wavelengths depending on the composition and grain size distribution, important effects to include in the disk models produced by *exovista*. For this reason, we use the Q_{abs} and Q_{sca} values calculated by Mie theory. We note that Mie theory can introduce "ringing" into the values of Q_{abs} and Q_{sca} , especially for grains similar in size to the wavelength. The plot below shows Q_{sca} vs wavelength for a 1 micron grain; the noticeable "ringing" is due to the assumed spherical geometry of the grain, which is almost certainly not representative of the real grain shape. If one considers a size distribution, this ringing goes away, but only if one resolves the size distribution with many grain sizes. More on this later...

For the SPF, we could either adopt the SPF calculated by Mie theory, or a linear combination of HG functions. We choose the latter for several reasons:

1. The SPF of dust grains is sensitive to grain shape, whereas Mie theory assumes a spherical geometry. Therefore it's an open question as to whether the Mie theory SPFs are representative of reality.
2. The SPFs of observed debris disks vary from disk to disk. To include this variation in *exoVista* while using Mie theory SPFs, we'd have to include some prescription for grain shape or allow *exoVista* to at least draw from widely different porosities/compositions. *ExoVista* does not currently have this capability.
3. Mie theory SPFs suffer greatly from "ringing" artifacts, where the SPF is bright at some scattering angles and faint at others, producing radial shadows in the disk. To average over these, a large number of grain sizes must be used, which slows down run time. The plot below and to the left compares the total flux from a face-on disk of 'organics' normalized to the stellar flux (i.e. disk contrast) vs wavelength using Mie Theory for all optical prescriptions with a grain size resolution (s/ds) ~ 60 (black line) to a grain size resolution ~ 5 (red line), keeping the full range of the size distribution constant (0.5 - 100 microns). This shows that reducing the # of grain sizes produces significant errors in the flux

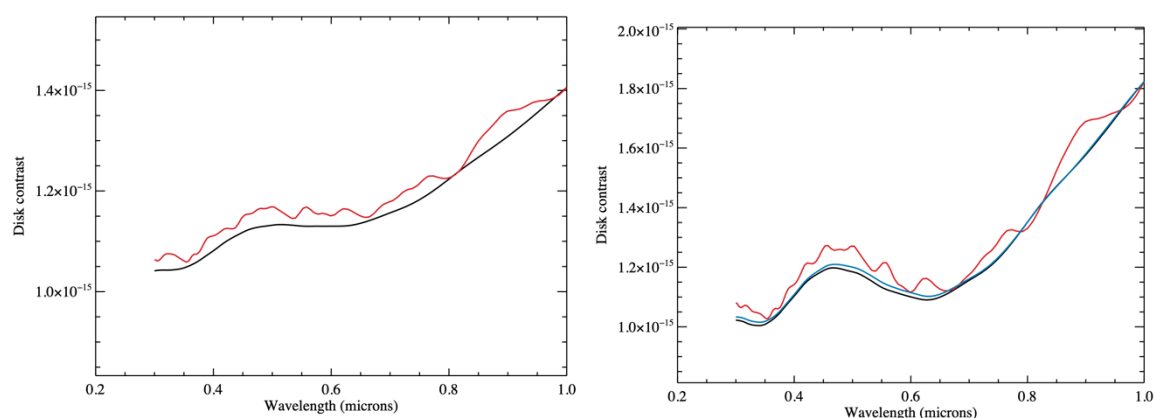
at a given scattering angle. The plot below and to the right shows the same calculations where we have substituted a HG SPF; the ringing in the Mie theory SPF is the cause of the disagreement on the left (not the ringing in Q_{sca}).



We therefore choose to bypass the Mie Theory SPF and adopt a linear combination of HG SPFs in *exoVista*, allowing us to reduce the number of grain sizes needed (which reduces run time) as well as randomize each disk's SPF by drawing different SPF asymmetry parameters. We note that while this choice is reasonable, it does impact the appearance and color of the simulated disks. Maybe most notably the Mie theory SPF changes with grain size, which thus impacts the color of the disks, especially if the grain size distribution in the disk changes. By using the same HG SPF for all grain sizes, only Q_{sca} integrated over the grain size distribution affects visible wavelength color. This generally leads to milder color variations in a disk. **Future updates to exovista may consider varying the HG SPF asymmetry parameters w/ grain size, but that is not currently included.**

Now we comment on the proper grain size resolution. In the test shown above, a grain size resolution $s/ds \sim 5$ was adequate for 'organics' when using HG SPFs, as they resulted in $\sim 1\%$ errors in the flux vs wavelength at a given scattering angle. Shown below and to the left is the same plot for astronomical silicates w/ a Dohnanyi size distribution. Here, in spite of using a HG SPF, $s/ds \sim 5$ (red line) is clearly inadequate, because in this case the Q_{sca} values greatly suffer from "ringing." The only way to average over these are to increase the number of grain sizes considered, which would greatly increase run time. As a work-around, *exoVista* uses pre-calculated Q_{sca} and Q_{abs} values that are integrated over a small range of grain sizes via .lqq files. Each .lqq file covers a small range of grain sizes and resolved the range with 100 individual sizes assuming a Dohnanyi distribution. The plot below and to the right shows the flux vs wavelength for 'astrosil' grains with a significantly non-Dohnanyi size distribution ($dN/ds \sim s^{-5.5}$), resolved at a higher size resolution of $s/ds \sim 60$ using

Mie theory on every grain size with (black line), $s/ds \sim 5$ with Mie theory (red line), and $s/ds \sim 5$ with pre-calculated .lqq files (blue line) with each .lqq file being sub-resolved into 100 grain sizes. Even in the case of an extremely non-Dohnanyi size distribution as shown here, the pre-calculated Q_{sca} and Q_{abs} in the .lqq files with $s/ds \sim 5$ produce results that differ by just $\sim 1\%$ from what is expected. Because the Mie theory calculations are done prior to creating the images, the run time of the code is further reduced. **We note that this improvement in run time and accuracy comes at the expense of complexity—if the user wants to change the grain size resolution or composition, the lqq files must be regenerated for that scenario.** If the user sticks with the default values, these will not need to be regenerated.



Finally, we touch on spectral resolution. The above curves were all calculated at a spectral resolution $R=300$. There are clearly no features at that high of spectral resolution that need to be resolved (except for the Mie theory ringing, which we have averaged over in our calculation of Q_{abs} and Q_{sca} during production of the .lqq files). We can greatly reduce runtime of the code if we save disk contrast vs wavelength at a lower spectral resolution. Testing astrosil, waterice, and organics, I find that $R=10$, interpolated back up to $R=300$ is sufficient to resolve disk spectral features. Below is a plot of disk contrast vs wavelength for a non-Dohnanyi distribution ($dN/ds \sim s^{-5.5}$) of organics calculated with $s/ds \sim 60$ at $R=300$ using Mie theory for each grain size (black), with $s/ds \sim 5$ at $R=300$ using .lqq files (red), and with $s/ds \sim 5$ at $R=10$ interpolated up to $R=300$ using .lqq files (blue dashed curve). The $R=10$ interpolated curve differs from the red curve by $\sim 1\%$ at most.

