Examples of Code Useful for Ring analysis

V1.0

Contents

How to find the ring/equator plane intercept

```
pro p2ralon,et,polera,poledec,RA,dec,radius,lon,length,phase,no_print=no_print

;Science analysis code for the Cassini Project was usually written in IDL using the IDL implementation of
;SPICE usually referred to as "ICY"


;This IDL procedure calculates the equator/ring plane intercept given a spacecraft/observer's position ;and
a pointing direction. Note that appropriate SPICE kernels must already be loaded.
;p2ralon is a contraction of "convert pointing to a ring/equator plane intercept radius and longitude"

; INPUTS
; et          ephemeris time (seconds)
; polera      right ascension of planetary pole (degrees)
; poledec     declination of planetary pole (degrees)
; RA          right ascension of camera boresight (degrees)
; dec         declination of camera boresight (degrees)

; OUTPUTS
; radius      radial location of camera boresight intercept with equator plane of planet (km)
; lon         longitude of camera boresight intercept with equator plane of planet (degrees)
; phase       phase at camera boresight intercept with equator plane (degrees)


sc=-32L                ;SPICE ID of Cassini
planet_id=699L         ;SPICE ID of Saturn

capN=(polera+90.0d0)*2.0d0*!dpi/360.0
capJ=(90.0d0-poledec)*2.0d0*!dpi/360.0

rot=dblarr(3,3)        ;3x3 matrix that transforms from inertial planet frame to J2000 frame

rot[0,0]=cos(capN)
rot[1,0]=-sin(capN)*cos(capJ)
rot[2,0]=sin(capN)*sin(capJ)
rot[0,1]=sin(capN)
rot[1,1]=cos(capN)*cos(capJ)
rot[2,1]=-cos(capN)*sin(capJ)
rot[0,2]=0.0
rot[1,2]=sin(capJ)
rot[2,2]=cos(capJ)

rho=dblarr(3)          ;camera boresight pointing unit vector in inertial planet frame
```

```
xyz=dblarr(3)                   ;apparent position of spacecraft in inertial planet frame (km)

xyz_J2000=dblarr(3)             ;apparent position of spacecraft in J2000 frame (km)
rho_J2000=dblarr(3)             ;camera boresight pointing unit vector in J2000 frame
xyz_rp=dblarr(3)

r=1.0d0*cos(2.0d0*!dpi*dec/360.0d0)
rho_J2000[0]=r*cos(2.0d0*!dpi*RA/360.0d0)
rho_J2000[1]=r*sin(2.0d0*!dpi*RA/360.0d0)
rho_J2000[2]=(sin(2.0d0*!dpi*dec/360.0d0))/1.0d0

trot=transpose(rot)  ;3x3 matrix that transforms from J2000 frame to inertial planet frame

d_rho=trot##rho_J2000

rho[0:2]=d_rho[0,0:2]

cspice_spkez,planet_id,et,'J2000','NONE',sc,state,light_time        ;SPICE call to obtain state vector of planet
;w.r.t spacecraft in J2000 frame (with light time correction)

;The state vector is obtained in this counterintuitive way in order to allow for the eventual correct
;implementation of the light time correction. Note that et is time at the spacecraft not the planet centre.
;Ultimately we require the position of the equator/ring plane at et-light time where the light time is the ;one
way light time to the intercept point NOT the centre of the planet.

xyz_J2000=-state[0:2]           ;apparent position of spacecraft in J2000 frame (km)

d_xyz=trot##xyz_J2000
xyz[0:2]=d_xyz[0,0:2]           ;apparent position of spacecraft in inertial planet frame (kilometers)

;if the spacecraft z coordinate is negative and the camera is pointing "down" that there will be no equator
;plane intercept

if xyz[2] lt 0.0 and rho[2] le 0.0 then begin
        radius=-1.0d0
        lon=-1.0d0
        length=-1.0d0
        return
endif

;if the spacecraft z coordinate is positive and the camera is pointing "up" then there will be no equator
;plane intercept

if xyz[2] gt 0.0 and rho[2] ge 0.0 then begin
        radius=-1.0d0
        lon=-1.0d0
```

```
        length=-1.0d0
        return
endif

length=abs(xyz[2]/rho[2])        ;the apparent distance to the equator plane intercept point (km)

xyz_rp=xyz+(length*rho)          ;the apparent position of the equator plane intercept point in the
;inertial planet frame

light_time=norm(xyz_rp-xyz)/299792.458d0     ;the light time from the spacecraft to the equator plane
;intercept point

;don't have to iterate for light time since location is fixed w.r.t the planet centre.

;now calculate the apparent position of the equator plane intercept point using the appropriate light time
;correction. Previous calculations used the one way light time to the planet centre

cspice_spkez,planet_id,et-light_time,'J2000','NONE',sc,state,ltime_dummy ;SPICE call to obtain state
;vector of planet w.r.t spacecraft in J2000 frame (with light time correction)

xyz_J2000=-state[0:2]            ;apparent position of spacecraft in J2000 frame - with light time
;correction (kilometers)
d_xyz=trot##xyz_J2000
xyz[0:2]=d_xyz[0,0:2]            ;apparent position of spacecraft in inertial planet frame (km)
;this bit allows for aberration effects assuming a solid, non-rotating disc, lying in the ring plane centred ;on
the planet.

length=abs(xyz[2]/rho[2])        ;apparent distance to the equator plane intercept point (km)

xyz_rp=xyz+(length*rho)          ;apparent position of the equator plane intercept point in the inertial
;planet frame

radius=norm(xyz_rp)              ;the radial distance from the centre of the planet to the equator plane
;intercept point (kilometers)

lon=(360.0d0*atan(xyz_rp[1],xyz_rp[0]))/(2.0d0*!dpi)   ;the longitude of the equator plane intercept
;point (degrees)

if lon lt 0.0 then lon=lon+360.0d0   ;making sure that the longitude is always positive


; to calculate phase


cspice_spkez,10L,et-light_time,'J2000','NONE',planet_id,state,ltime_dummy     ;SPICE call to obtain state
;vector of the Sun w.r.t the planet in J2000 frame
```

```
xyz_J2000_sun=state[0:2]              ;position of Sun in J2000 frame
d_xyz=trot##xyz_J2000_sun
xyz_sun=dblarr(3)                     ;position of Sun in inertial planet frame
xyz_sun[0:2]=d_xyz[0,0:2]

object_sun=xyz_sun-xyz_rp             ;position of the Sun w.r.t to equator plane intercept point
object_sc=-rho                        ;position of spacecraft w.r.t the equator plane intercept point

phase=cspice_vsep(object_sc,object_sun)*360.0d0/(2.0d0*!dpi)        ;solar phase angle at ring plane
;intercept point

elevxy=sqrt((rho[0]*rho[0])+(rho[1]*rho[1]))
elevation=(360.0d0/(2.0d0*!dpi))*atan(-rho[2],elevxy)
if keyword_set(no_print) eq 0 then begin

        print,' '
        print,'Phase = '+strtrim(string(phase),2) +' degrees (at ring plane intercept)'
        print,'Distance to ring plane intercept = '+strtrim(string(length),2)+' km'
        print,'Elevation at ring plane intercept = '+strtrim(string(elevation),2)+' deg'
        print,' '
endif

return
end
```