Derived Quantities

January 2017
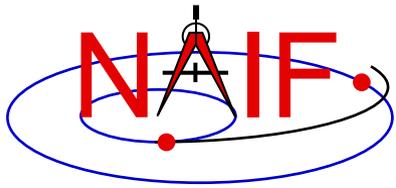
What are Derived Quantities?

- **Derived quantities are produced using data from kernels.**
  - **These are the primary reason that SPICE exists!**

- **The SPICE Toolkit contains many routines that assist with the computations of derived quantities.**
  - Some are fairly low level, some are quite high level.
  - More are being added as time permits.

- **Examples follow on the next several pages.**
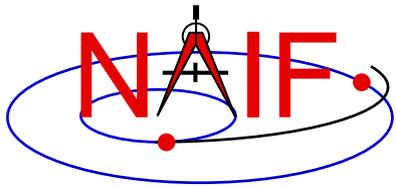
High-level Geometric Quantities

- **Illumination angles (phase, incidence, emission)**
  - **ILUMIN***
- **Illumination angles with any ephemeris object as the source**
  - **ILLUMG**
- **Subsolar point**
  - **SUBSLR***
- **Subobserver point**
  - **SUBPNT***
- **Surface intercept point**
  - **SINCPT***
- **Longitude of the sun (L_s), an indicator of season**
  - **LSPCN**
- **Phase angle between body centers**
 - **PHASEQ**
- **Terminator on an ellipsoid**
  - **EDTERM**

* These routines supercede the now deprecated routines ILLUM, SUBSOL, SUBPT and SRFXPT

Geometric Condition

- **Ray in field-of-view?**
  - **FOVRAY**

- **Ephemeris object within field-of-view?**
  - **FOVTRG**

- **Determine occultation or transit condition**
 - **OCCULT**
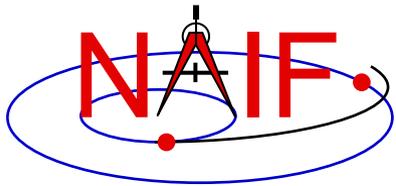
Geometric Event Finder

- **Most SPICE routines are used to determine a quantity at a specified time.**

- **The SPICE Geometry Finder (GF) subsystem takes the opposite approach: find times, or time spans, when a specified geometric condition or event occurs.**
  - **This is such a large topic that a separate tutorial ("geometry_finder") has been written for it.**
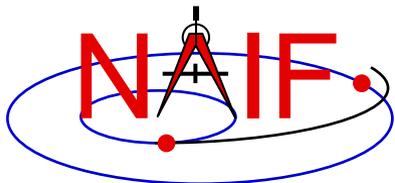
Geometric Objects

Function   Routine

- **Ellipsoids**
    - **nearest point**   – `NEARPT, SUBPNT, DNEARP`
    - **surface ray intercept**   – `SURFPT, SINCPT`
    - **surface normal**   – `SURFNM`
    - **limb**   – `EDLIMB`
    - **slice with a plane**   – `INELPL`
    - **altitude of ray w.r.t. to ellipsoid**   – `NPEDLN`
- **Planes**
    - **intersect ray and plane**   – `INRYPL`
- **Ellipses**
    - **project onto a plane**   – `PJELPL`
    - **find semi-axes of an ellipse**   – `SAELGV`

Coordinate Transformation   Routine

- Transform state vector between two coordinate systems
  - XFMSTA   **General purpose API**

- Latitudinal to/from Rectangular

- Planetographic to/from Rectangular

- R.A. Dec to/from Rectangular

- Geodetic to/from Rectangular

- Cylindrical to/from Rectangular

- Spherical to/from Rectangular

  - LATREC RECLAT

  - PGRREC RECPGR

  - RADREC RECRAD   **Single purpose APIs**

  - GEOREC RECGEO

  - CYLREC RECCYL

  - SPHREC RECSPH

Vectors

- **Function**
  - **<v,w>**
  - **v** x **w**
  - **v/|v|**
  - **v** x **w** / | **v** x **w**|
  - **v + w**
  - **v - w**
  - a**v** [**+** b**w** [**+** c**u**]]
  - angle between **v** and **w**
  - **|v|**

- **Routine**
  - **VDOT,   DVDOT**
  - **VCROSS,   DVCRSS**
  - **VHAT,   DVHAT**
  - **UCROSS,   DUCRSS**
  - **VADD,   VADDG**
  - **VSUB,   VSUBG**
  - **VSCL,   [VLCOM, [VLCOM3]]**
  - **VSEP**
  - **VNORM**
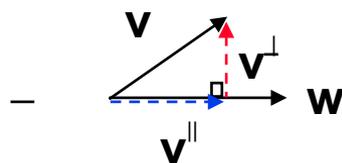
  - **VPROJ,   VPERP**
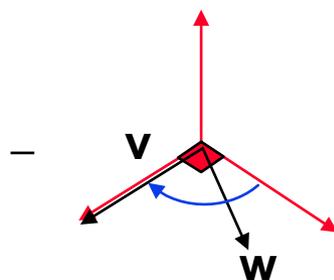
  - **TWOVEC,   FRAME**
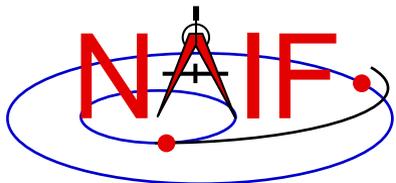
Matrices

Selected Matrix-Vector Linear Algebra Routines

- **Function**
  - **M x v**
  - **M x M**
  - **M^t x v**
  - **M^t x M**
  - **M x M^t**
  - **v^t x M x v**
  - **M^t**
  - **M^{-1}**

- **Routine**
  - **MXV**
  - **MXM**
  - **MTXV**
  - **MTXM**
  - **MXMT**
  - **VTMV**
  - **XPOSE**
  - **INVERT, INVSTM**

M = Matrix

V = Vector

**X** = Multiplication

T = Transpose
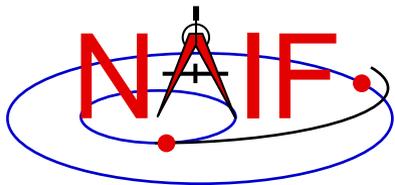
Matrix Conversions


Navigation and Ancillary Information Facility

Function

Routines

Euler angles
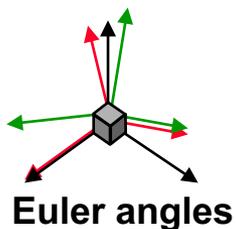
← Transform between →

$$\begin{matrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{matrix}$$

3x3 rotation matrix

— **EUL2M, M2EUL**

Euler angles and Euler angle rates
or
rotation matrix and angular velocity vector
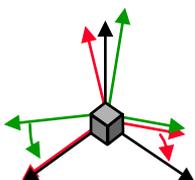
← Transform between →

$$\begin{matrix} a_x & a_y & a_z & & & \\ b_x & b_y & b_z & & 0 & \\ c_x & c_y & c_z & & & \\ \alpha_x & \alpha_y & \alpha_z & a_x & a_y & a_z \\ \beta_x & \beta_y & \beta_z & b_x & b_y & b_z \\ \gamma_x & \gamma_y & \gamma_z & c_x & c_y & c_z \end{matrix}$$

6x6 state transformation matrix

— **EUL2XF, XF2EUL**
  RAV2XF, XF2RAV

Rotation axis and angle
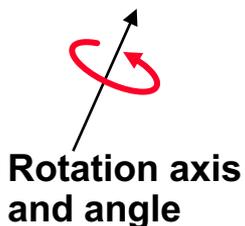
← Transform between →

$$\begin{matrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{matrix}$$

3x3 rotation matrix

— **RAXISA, AXISAR**
  ROTATE, ROTMAT

(Q_0, Q_1, Q_2, Q_3)

SPICE Style Quaternion

← Transform between →

$$\begin{matrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{matrix}$$

3x3 rotation matrix

— **Q2M, M2Q**

Derived Quantities
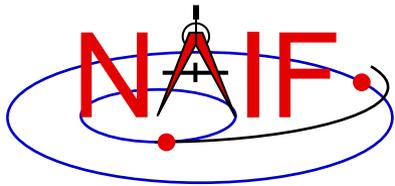
Navigation and Ancillary Information Facility

Coordinate Transformation

Routine

– **Transform state vector between two coordinate systems**	– `XFMSTA` **General purpose API**

– **Latitudinal to/from Rectangular**	– `DRDLAT` `DLATDR`	
– **Planetographic to/from Rectangular**	– `DRDPGR` `DPGRDR`	
– **R.A. Dec to/from Rectangular**	– `DRDLAT` `DLATDR`	
– **Geodetic to/from Rectangular**	– `DRDGEO` `DGEODR`	**Single purpose Jacobian matrix APIs**
– **Cylindrical to/from Rectangular**	– `DRDCYL` `DCYLDR`	
– **Spherical to/from Rectangular**	– `DRDSPH` `DSPHDR`	
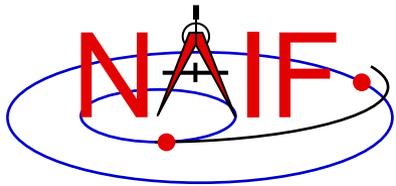
Examples of Velocity Coordinate Transformations

This example is for rectangular to spherical

- ## Using full state vector transformation API

  CALL SPKEZR (TARG, ET, REF, CORR, OBS, STATE, LT)

 CALL XFMSTA (STATE, 'RECTANGULAR', 'SPHERICAL', ' ' , OUTSTATE)

- ## Using velocity-only (Jacobian) APIs

  – Transform velocities from rectangular to spherical coordinates using the SPICE Jacobian matrix routines. The SPICE calls that implement this computation are:

  CALL SPKEZR (TARG, ET, REF, CORR, OBS, STATE, LT)

  CALL DSPHDR (STATE(1), STATE(2), STATE(3), JACOBI)

  CALL MXV (JACOBI, STATE(4), SPHVEL)

  – After these calls, the vector SPHVEL contains the velocity in spherical coordinates: specifically, the derivatives

 (d (r) / dt, d (colatitude) / dt, d (longitude) /dt)

  – Caution: coordinate transformations often have singularities, so derivatives may not exist everywhere.

    » **Exceptions are described in the headers of the SPICE Jacobian matrix routines.**

    » **SPICE Jacobian matrix routines signal errors if asked to perform an invalid computation.**

- **On the next several pages we present examples of using some of the "derived quantity" APIs.**

- **Given the direction of an instrument boresight in a body-fixed frame, return the illumination angles (incidence, phase, emission) at the surface intercept on a tri-axial ellipsoid**
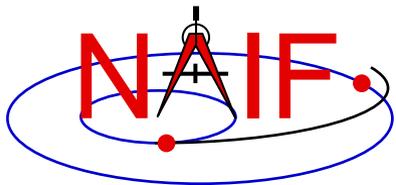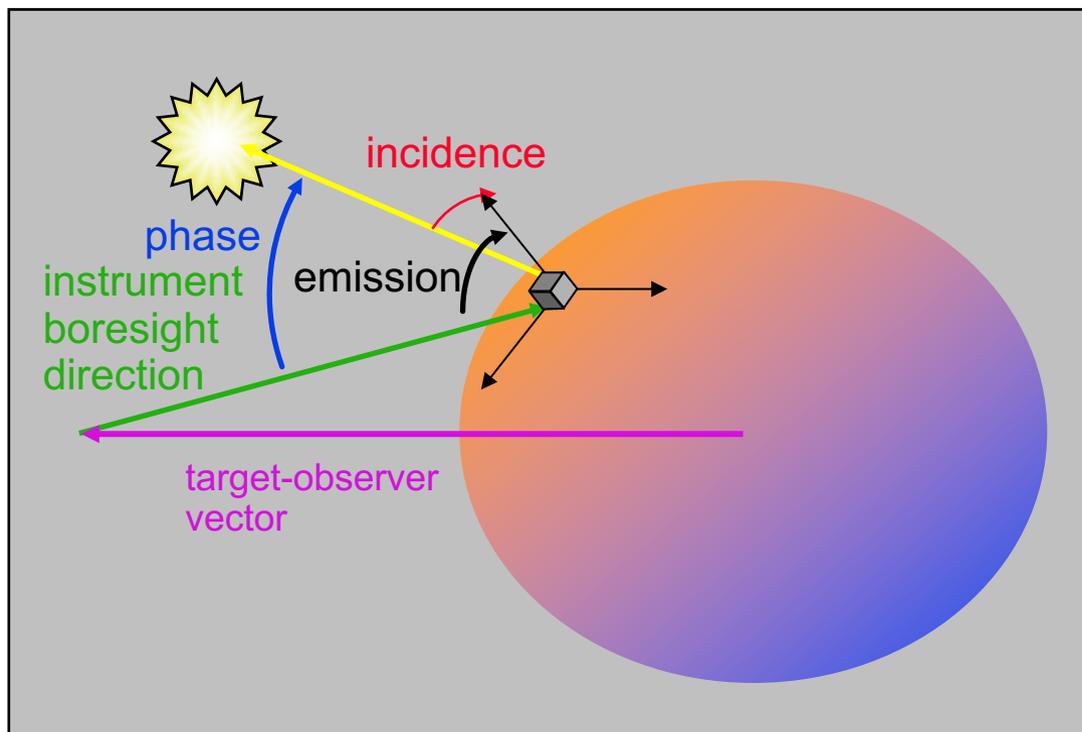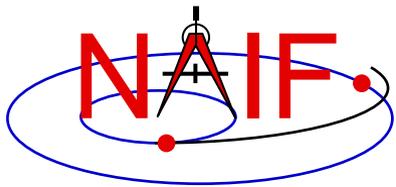
Computing Illumination Angles

- **CALL <u>GETFOV</u> to obtain boresight direction vector**

- **CALL <u>SINCPT</u> to find intersection of boresight direction vector with surface**

- **CALL <u>ILUMIN</u> to determine illumination angles**
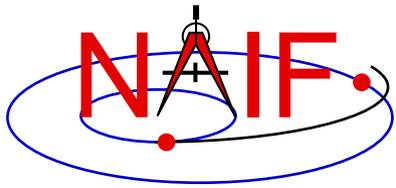
**Navigation and Ancillary Information Facility**

- **Determine the intersection of the apparent line of sight vector between Earth and Cassini with Saturn's ring plane and determine the distance of this point from the center of Saturn.**
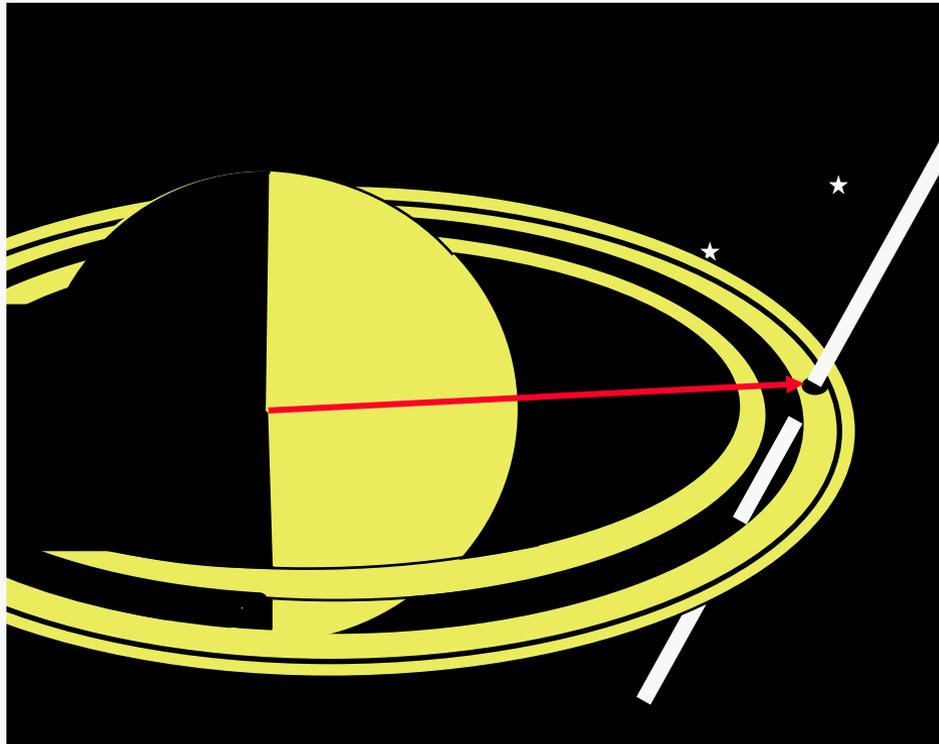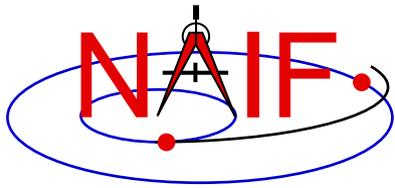
Navigation and Ancillary Information Facility

> This computation is for the reception case; radiation is received at the earth at a given epoch "ET".
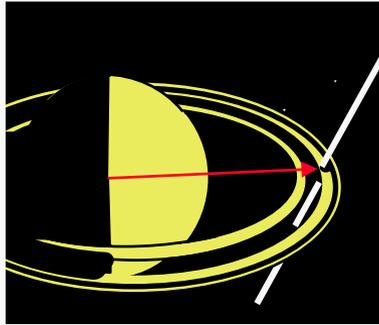
This simplified computation ignores the difference between the light time from Saturn to the earth and the light time from the ring intercept point to the earth.

The position and orientation of Saturn can be re-computed using the light time from earth to the intercept; the intercept can be re-computed until convergence is attained.

- **CALL SPKEZR to get light time corrected position of spacecraft as seen from earth at time ET in J2000 reference frame SCVEC.**

- **CALL SPKEZR to get light time corrected position of center of Saturn at time ET as seen from earth in J2000 frame SATCTR.**

- **CALL PXFORM to get rotation from Saturn body-fixed coordinates to J2000 at light time corrected epoch. The third column of this matrix gives the pole direction of Saturn in the J2000 frame SATPOL.**

- **CALL NVP2PL and use SATCTR and SATPOL to construct the ring plane RPLANE.**

- **CALL INRYPL to intersect the earth-spacecraft vector SCVEC with the Saturn ring plane RPLANE to produce the intercept point X.**

- **CALL VSUB to get the position of the intercept with respect to Saturn XSAT (subtract SATCTR from X) and use VNORM to get the distance of XSAT from the center of Saturn.**
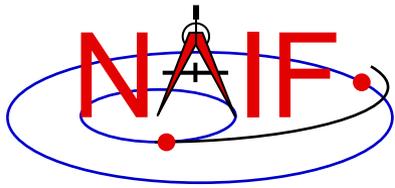
Navigation and Ancillary Information Facility
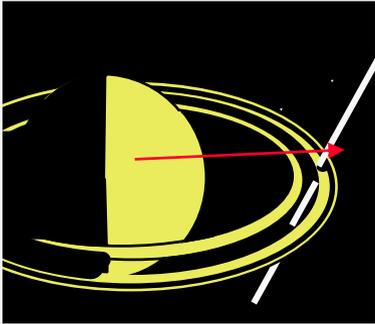
An alternate approach

- Create a dynamic frame with one axis pointing from earth to the light time corrected position of the Cassini orbiter. Use the CN correction for this position vector. (This gives us a frame in which the direction vector of interest is constant.)

- Temporarily change the radii of Saturn to make the polar axis length 1 cm and the equatorial radii 1.e6 km. This can be done either by editing the PCK or by calling **BODVCD** to fetch the original radii, then calling **PDPOOL** to set the kernel pool variable containing the radii to the new values. This flat ellipsoid will be used to represent the ring plane.

- Use **SINCPT** to find the intercept of the earth-Cassini ray with the flat ellipsoid. Use the CN correction. SINCPT returns both the intercept in the IAU_SATURN frame and the earth-intercept vector. Use **VNORM** to get the distance of the intercept from Saturn's center.

- Restore the original radii of Saturn. If PDPOOL was used to update the radii in the kernel pool, use **PDPOOL** again to restore the radii fetched by BODVCD.
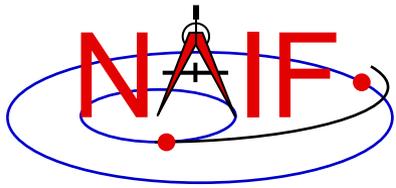
Derived Quantities

Navigation and Ancillary Information Facility

- **Determine when the spacecraft will be occulted by an object (such as a natural satellite) as seen from an observer (such as earth).**

Direction to the observer

Spacecraft motion

A body such as a satellite
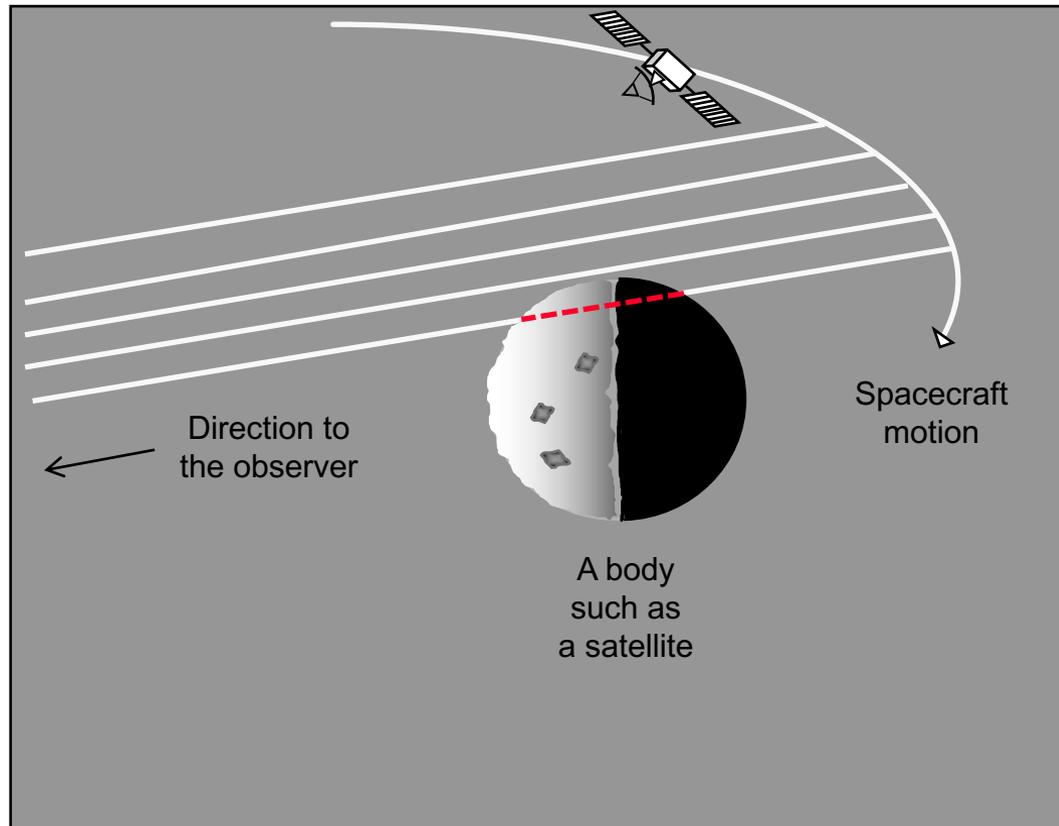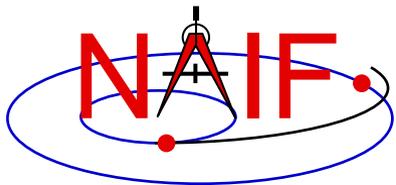
Find Occultation Ingress/Egress

Direction to the observer

A body such as a satellite

Spacecraft motion

- **Select a start epoch, stop epoch and step size.**
  - Start and stop epochs can bracket multiple occultation events
  - Step size should be smaller than the shortest occultation duration of interest, and shorter than the minimum interval between occultation events that are to be distinguished, but large enough to solve problem with reasonable speed.
  - Insert search interval into a SPICE window. This is the "confinement window."
- **CALL GFOCLT to find occultations, if any. The time intervals, within the confinement window, over which occultations occur will be returned in a SPICE window.**
  - GFOCLT can treat targets as ellipsoids or points (but at least one must be an ellipsoid).
  - GFOCLT can search for different occultation or transit geometries: full, partial, annular, or "any."