- **One reads this file the same as for any other SPK file**
  - **Use the name or NAIF ID of the antenna, observatory or rover as the “target” or “observer” in an SPK reader argument list**
  - **Also requires use of a SPICE PCK file if you request vectors to be returned in an inertial frame such as J2000**
    - » **Needed to rotate body-fixed vectors to the J2000 frame**

Navigation and Ancillary Information Facility

Using SPK Files

Retrieving Position or State Vectors

Navigation and Ancillary Information Facility

- To retrieve position or state vectors of ephemeris objects one normally needs two kinds of SPICE kernels
  - Ephemeris kernel(s) (SPK)
  - Leapseconds kernel (LSK)
    - Used to convert between Coordinated Universal Time (UTC) and Barycentric Dynamical Time (TDB, also called Ephemeris Time, ET)
- Retrieving ephemeris data from an SPK file is usually called "reading" the file
  - This term is not very accurate since the SPK "reader" software also performs interpolation, and may chain together data from multiple sources, do frame transformations and perform aberration corrections
- State and position vectors retrieved from an SPK file by the SPK "reader" routines are of the form:
  - X,Y, Z, dX, dY, dZ for a state vector (km, km/sec)
  - X, Y, Z for a position vector (km)

N IF

Retrieving a State Vector

Navigation and Ancillary Information Facility

Fortran syntax used here

Initialization…typically done once per program execution

Tell your program which SPICE files to use ("loading" files)

CALL FURNSH ('spk_file_name')
CALL FURNSH ('leapseconds_file_name')

It's better to replace these two calls with a single call to a "furnsh kernel" containing the names of all kernel files to load.

Loop... do as many times as you need to

Convert UTC time to ephemeris time (TDB), if needed

CALL STR2ET ('utc_string', *tdb*)

Retrieve state vector from the SPK file at your requested time

CALL SPKEZR (target, tdb, 'frame', 'correction', observer, *state, light time*)

inputs outputs **Now use the returned state vector in other SPICE routines to compute observation geometry of interest.**

Arguments of SPKEZR - 1

INPUTS

- **TARGET* and OBSERVER*: Character names or NAIF IDs for the end point and origin of the state vector (Cartesian position and velocity vectors) to be returned.**
  - **The position component of the requested state vector points from observer to target.**
- **TDB: The time at the observer at which the state vector is to be computed. The time system used is Ephemeris Time (ET), now generally called Barycentric Dynamical Time (TDB).**
- **FRAME: The SPICE name for the reference frame in which the output state vector is to be given. SPK software will automatically convert ephemeris data to the frame you specified (if needed). SPICE must**

know the named frame. If it is not a built-in frame SPICE must have sufficient data at run-time to construct it.

* Character names work for the target and observer inputs only if built into SPICE or if registered using the SPICE ID-body name mapping facility. Otherwise use the SPICE numeric ID in quotes, as a character string.

Arguments of SPKEZR - 2

- **CORRECTION: Specification of what kind of aberration correction(s), if any, to apply in computing the output state vector.**
  - **Use ‘LT+S’ to obtain the apparent state of the target as seen by the observer. ‘LT+S’ invokes light time and stellar aberration corrections. (‘CN+S’ is better in some cases.)**
  - **Use ‘NONE’ to obtain the uncorrected (aka “geometric”) state, as given by the source SPK file or files.**

See the header for subroutine SPKEZR, the document SPK Required Reading, or the “Fundamental Concepts” tutorial for details. See the backup charts for examples of aberration correction magnitudes.

OUTPUTS

- *STATE*: This is the Cartesian state vector you requested. Contains 6 components: three for position (x,y,z) and three for velocity (dx, dy, dz) of the target with respect to the observer. The position component of the state vector points from the *observer* to the *target*.
- *LIGHT TIME*: The one-way light time between the (optionally aberration-corrected) position of target and the geometric position of the observer at the specified epoch.

LT + S = light time plus stellar aberration
CN + S = converged Newtonian light time plus stellar aberration

A Simple Example of Retrieving a State Vector

Navigation and Ancillary Information Facility

Initialization - typically do this just once per program execution

```
CALL FURNSH ( 'NAIF0010.TLS' )
FURNSH ( 'HUYGENS_3_MERGE.BSP' )
```

two calls — It's better to replace these with a single call to **CALL** FURNSH using a "furnsh kernel" containing the names of all kernel files to load.

Repeat in a loop if/as needed to solve your particular problem

NAIF

```
CALL STR2ET ('2004 NOV 21 02:40:21.3', TDB ) CALL SPKEZR
('TITAN', TDB, 'J2000', 'LT+S', 'HUYGENS PROBE',
STATE,  LT )
```

(Insert additional code here to make derived computations such as spacecraft sub-latitude and longitude, lighting angles, etc. Use more SPICE subroutines to help.)

In this example we get the state (STATE) of Titan as seen from the Huygens probe at the UTC epoch 2004 NOV 21 02:40:21.3. The state vector is returned in the J2000 inertial reference frame (which in SPICE is the same as the ICRF frame) and has been corrected for both light time and stellar aberration (LT+S). The oneway light time (LT) is also returned.

A SPICE leapseconds file (NAIF0010.TLS) is used, as is a SPICE ephemeris file (HUYGENS_3_MERGE.BSP) containing ephemeris data for the Huygens probe (-150), Saturn barycenter (6), Saturn mass center (699), Saturn's satellites (6xx) and the sun (10), relative to the solar system barycenter.

Retrieving a Position Vector

- **SPKPOS is the position-only analog of SPKEZR**

– The arguments of SPKPOS are identical to those of SPKEZR, except that SPKPOS returns a 3-component position vector instead of a 6-component state vector
– SPKPOS executes more quickly than SPKEZR when stellar aberration corrections are used
– SPKPOS can be used when reference frame transformations of velocity are not possible due to absence of C-kernel angular velocity data

A Slightly More Complex Example - 1
Kernel Data Needed

- **To get state vectors referenced to a non-inertial reference frame, or when the data within the SPK file are provided in a non-inertial frame, typically more kernels will be needed.**
  – To get the state of an object relative to a body in the body's IAU body-fixed reference frame you'll need:

N IF

» PCK file containing orientation data for the body

» SPK(s) for the object and body

» LSK

– To get the state of an object in a spacecraft-fixed reference frame you'll need:

» FK, CK and SCLK for the spacecraft

» SPK(s) for the spacecraft and object

» LSK

A Slightly More Complex Example - 2 Retrieving a State Vector

Obtain the state of Titan relative to Huygens Probe in the Titan body-fixed reference frame

Initialization...typically once per program execution

Tell your program which SPICE files to use (“loading” files)

```
CALL FURNSH  ('HUYGENS_MERGED_SPK.BSP')
CALL FURNSH  ('NAIF0010.TLS')
CALL FURNSH  ('NAIF0010.TPC')
```

It’s better to replace these three calls with a single call to a “furnsh kernel” containing the names of all kernel files to load.

Loop... do as many times as you need

Convert UTC time to ephemeris time (TDB), if needed

```
CALL STR2ET ( '2004 NOV 21 02:40:21.3', TDB)
```

Get state vector from SPK file at requested time, in planet’s IAU body-fixed frame

```
CALL SPKEZR ('TITAN', TDB, 'IAU_TITAN', 'LT+S', 'HUYGENS
   PROBE', STATE, LT)
```

```
(Insert additional code here to make derived computations such as
spacecraft sub-latitude and longitude, lighting angles, etc. Use more
SPICE subroutines to help.
```

Constant-velocity objects:

Additional SPK State Computation APIs

- The SPK subsystem contains routines for computing the state of an ephemeris object with respect to a fixed point or one moving with constant, non-zero velocity, in a specified reference frame.
  - The point may act as either the target or the observer.
  - The center of motion of the point must be an ephemeris object.
- The SPK routines providing this capability are:
  - SPKCPT (SPK, constant position target)
  - SPKCPO (SPK, constant position observer)
  - SPKCVT (SPK, constant velocity target)
  - SPKCVO (SPK, constant position observer)
- These routines may provide a convenient alternative to creating SPK files for surface points:
  - when there’s a need to compute the locations on the fly

- **You can subset an SPK, or merge two or more SPKs**
  - **The subset or merge may be keyed off of objects, or time, or both**

- **You can read data from just one, or many* SPK files in your application program**
  - **Don't forget the precedence rule: data in a later loaded file take precedence over data from an earlier loaded file**

- **You can convert an SPK that is in non-native binary format to native binary format if you need to add data or comments**

* The allowed number of simultaneously loaded DAF-based files is set to 5000 in N65 Toolkits. “DAF” is Double Precision Array File.

Understanding an SPK File

- **The SPK producer should have provided descriptive meta-data inside an SPK file, in the “comment area”**
  - **The comments should say when/why/how and for what purpose the file was made**
  - **Additional useful information could also be provided by the producer**
    - » **Example: when and why any data gaps are present**

- **These comments may be extracted or viewed using an API (subroutine) or a SPICE utility program.**
  - **APIs: SPC…**
  - **Utility program: commnt -r**

SPK Utility Programs

Navigation and Ancillary Information Facility

- **The following SPK utility programs are included in the Toolkit:**

BRIEF	**summarizes coverage for one or more SPK files**
SPACIT	**generates segment-by-segment summary of an SPK file**
COMMNT	**reads, appends, or deletes comments in an SPK file**
MKSPK	**converts ephemeris data provided in a text file into an SPK file**
SPKDIFF	**compares two SPK files**
SPKMERGE	**subsets or merges one or more SPK files**

- **These additional SPK utility programs are provided on the NAIF Web site (http://naif.jpl.nasa.gov/naif/utilities.html)**

SPY	**validates, inspects, and analyses SPK files**
PINPOINT	**creates an SPK file for fixed locations (ground stations, etc)**

BSPIDMOD	**alters body IDs in an SPK file**
DAFMOD	**alters body or frame IDs in an SPK file**
DAFCAT	**concatenates together SPK files**
BFF	**displays binary file format of an SPK file**
BINGO	**converts SPK files between big- and little-endian formats**

Summarizing an SPK File - 1

Navigation and Ancillary Information Facility

- A summary of the contents and time coverage of an SPK file can be made using the SPICE Toolkit utility ***"brief"***
  - See the brief User's Guide for details

```
% brief 070413BP_SCPSE_07097_07121.bsp



Summary for: 070413BP_SCPSE_07097_07121.bsp

Bodies: CASSINI (-82)              PLUTO BARYCENTER (9)     TETHYS (603)
        MERCURY BARYCENTER (1)     SUN (10)                 DIONE (604)
        VENUS BARYCENTER (2)       MERCURY (199)            RHEA (605)
        EARTH BARYCENTER (3)       VENUS (299)              TITAN (606)
        MARS BARYCENTER (4)        MOON (301)               HYPERION (607)
        JUPITER BARYCENTER (5)     EARTH (399)              IAPETUS (608)
        SATURN BARYCENTER (6)      MARS (499)               PHOEBE (609)
        URANUS BARYCENTER (7)      MIMAS (601)              SATURN (699)
        NEPTUNE BARYCENTER (8)     ENCELADUS (602)
```

Note, the default time system is

ET, not UTC!

```
Start of Interval (ET)              End of Interval (ET)
-----------------------------       -----------------------------

2007 APR 07 16:22:23.000            2007 MAY 01 09:34:03.000
```

Summarizing an SPK File - 2

- Use of the “-c” option in brief will show you the center of motion for each object
  - This is often useful in diagnosing an SPK chaining problem
    - » See the “Problems” section at the end of this tutorial for more information

```
% brief -c 070413BP_SCPSE_07097_07121.bsp

Bodies: CASSINI (-82) w.r.t. SATURN BARYCENTER (6)
        MERCURY BARYCENTER (1) w.r.t. SOLAR SYSTEM BARYCENTER (0)
        VENUS BARYCENTER (2) w.r.t. SOLAR SYSTEM BARYCENTER (0)
        EARTH BARYCENTER (3) w.r.t. SOLAR SYSTEM BARYCENTER (0)
        MARS BARYCENTER (4) w.r.t. SOLAR SYSTEM BARYCENTER (0)
        JUPITER BARYCENTER (5) w.r.t. SOLAR SYSTEM BARYCENTER (0)
        SATURN BARYCENTER (6) w.r.t. SOLAR SYSTEM BARYCENTER (0)
        URANUS BARYCENTER (7) w.r.t. SOLAR SYSTEM BARYCENTER (0)
        NEPTUNE BARYCENTER (8) w.r.t. SOLAR SYSTEM BARYCENTER (0)
        PLUTO BARYCENTER (9) w.r.t. SOLAR SYSTEM BARYCENTER (0)
        SUN (10) w.r.t. SOLAR SYSTEM BARYCENTER (0)
        MERCURY (199) w.r.t. MERCURY BARYCENTER (1)
        VENUS (299) w.r.t. VENUS BARYCENTER (2)
        MOON (301) w.r.t. EARTH BARYCENTER (3)
        EARTH (399) w.r.t. EARTH BARYCENTER (3)
        MARS (499) w.r.t. MARS BARYCENTER (4)
```

```
MIMAS (601) w.r.t. SATURN BARYCENTER (6)
ENCELADUS (602) w.r.t. SATURN BARYCENTER (6)
TETHYS (603) w.r.t. SATURN BARYCENTER (6)
DIONE (604) w.r.t. SATURN BARYCENTER (6)
RHEA (605) w.r.t. SATURN BARYCENTER (6)
TITAN (606) w.r.t. SATURN BARYCENTER (6)
HYPERION (607) w.r.t. SATURN BARYCENTER (6)
IAPETUS (608) w.r.t. SATURN BARYCENTER (6)
PHOEBE (609) w.r.t. SATURN BARYCENTER (6)
SATURN (699) w.r.t. SATURN BARYCENTER (6)
Start of Interval (ET)              End of Interval (ET)
-----------------------------       -----------------------------
2007 APR 07 16:22:23.000            2007 MAY 01 09:34:03.000
```

Summarizing an SPK File - 3

- A detailed summary can be made using the Toolkit utility "SPACIT"
- See the SPACIT User's Guide for details

```
Summary for SPK file: sat240.bsp Leapseconds File    :
/kernels/gen/lsk/leapseconds.ker Summary Type        :
Entire File

--------------------------------------------------------------------------------
   Segment ID     : SAT240
   Target Body    : Body 601, MIMAS
   Center Body    : Body 6, SATURN BARYCENTER
   Reference frame: Frame 1, J2000
   SPK Data Type  : Type 3
      Description : Fixed Width, Fixed Order Chebyshev Polynomials: Pos, Vel
   UTC Start Time : 1969 DEC 31 00:00:00.000
   UTC Stop Time  : 2019 DEC 02 00:00:00.000
```

Summarizing an SPK File - 4


```
   ET Start Time    : 1969 DEC 31 00:00:41.183
   ET Stop time     : 2019 DEC 02 00:01:05.183
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
   Segment ID       : SAT240
   Target Body      : Body 602, ENCELADUS
   Center Body      : Body 6, SATURN BARYCENTER
   Reference frame: Frame 1, J2000
   SPK Data Type    : Type 3
      Description : Fixed Width, Fixed Order Chebyshev Polynomials: Pos, Vel
   UTC Start Time : 1969 DEC 31 00:00:00.000
   UTC Stop Time  : 2019 DEC 02 00:00:00.000
   ET Start Time  : 1969 DEC 31 00:00:41.183
   ET Stop time   : 2019 DEC 02 00:01:05.183

--------------------------------------------------------------------------------:
                                        :
```

(This is a partial output; not all data could be displayed on this chart)

Summarizing an SPK at the API Level

- **Call SPKOBJ to find the set of objects for which a specified SPK provides data.**
  - **INPUT: an SPK file name and initialized SPICE integer "Set" data structure. The set may optionally contain ID codes obtained from previous calls.**

– OUTPUT: the input set, to which have been added (via set union) the ID codes of objects for which the specified SPK provides coverage.

```
CALL SPKOBJ ( SPK, IDSET )
```

- Call SPKCOV to find the window of times for which a specified SPK file provides coverage for a specified body:
  - INPUT: an SPK file name, body ID code and initialized SPICE double precision “Window” data structure. The window may optionally contain coverage data from previous calls.
  - OUTPUT: the input window, to which have been added (via window union) the sequence of start and stop times of segment coverage intervals of the specified SPK, expressed as seconds past J2000 TDB.

```
CALL SPKCOV ( SPK, IDCODE, COVER )
```

- See the headers of these routines for example programs.
- Also see the CELLS, SETS and WINDOWS Required Reading for background information on these SPICE data types.

Additional Information on SPK

Navigation and Ancillary Information Facility

- For more information about SPK, look at the following:
  - The on-line (full) SPK tutorial
  - Most Useful Routines document
  - SPK Required Reading document
  - Headers of the subroutines mentioned
  - Using Frames tutorial
  - BRIEF and SPKDIFF User's Guides
- Related documents:
  - NAIF_IDS Required Reading
  - Frames Required Reading
  - Time Required Reading

– Kernel Required Reading

Backup

- Problems Using SPK Files
- Don’t Mix Planet Ephemerides
- Effect of Aberration Corrections
- Examples of Retrieving State Vectors
- SPK File Structure

Problems Using SPK Files - 1

- **The file, or files, you loaded do not contain data for <u>both</u> your target and observer bodies**
  - **You may have loaded the wrong file, or assumed the file contains data that it doesn't**
  - **You may not have loaded all the files needed**
- **The file, or files, you loaded do not cover the <u>time</u> at which you requested a state vector**
  - **This could occur if you've been given a file coverage summary in calendar ET form and you mistook this for UTC**

    (ET = UTC + DELTAET, where DELTAET is about 65 seconds as of 9/11)
  - **This could occur if you are requesting a light-time corrected state and the SPK files being used do not have data at the time that is one-way lighttime away* from your ET epoch of interest**
    - » *** Earlier, for the receive case; later, for the transmission case**
- **In the above situations you'll get an error message like the following:**

```
SPICE(SPKINSUFFDATA) --
Insufficient ephemeris data has been loaded to
compute the state of xxx relative to yyy.
```

Problems Using SPK Files - 2

- **You have requested aberration-corrected states but the file, or files, you loaded do not contain sufficient data to relate both your target and observer bodies back to the solar system barycenter, which is required for this calculation.**
  - **You may not have loaded all the files needed**
  - **You may have assumed the file contains data that it doesn't**
- **In the above situations you'll get an error message like the following:**

```
SPICE(SPKINSUFFDATA) - -
Insufficient ephemeris data has been loaded to compute
the state of xxx relative to yyy.
```

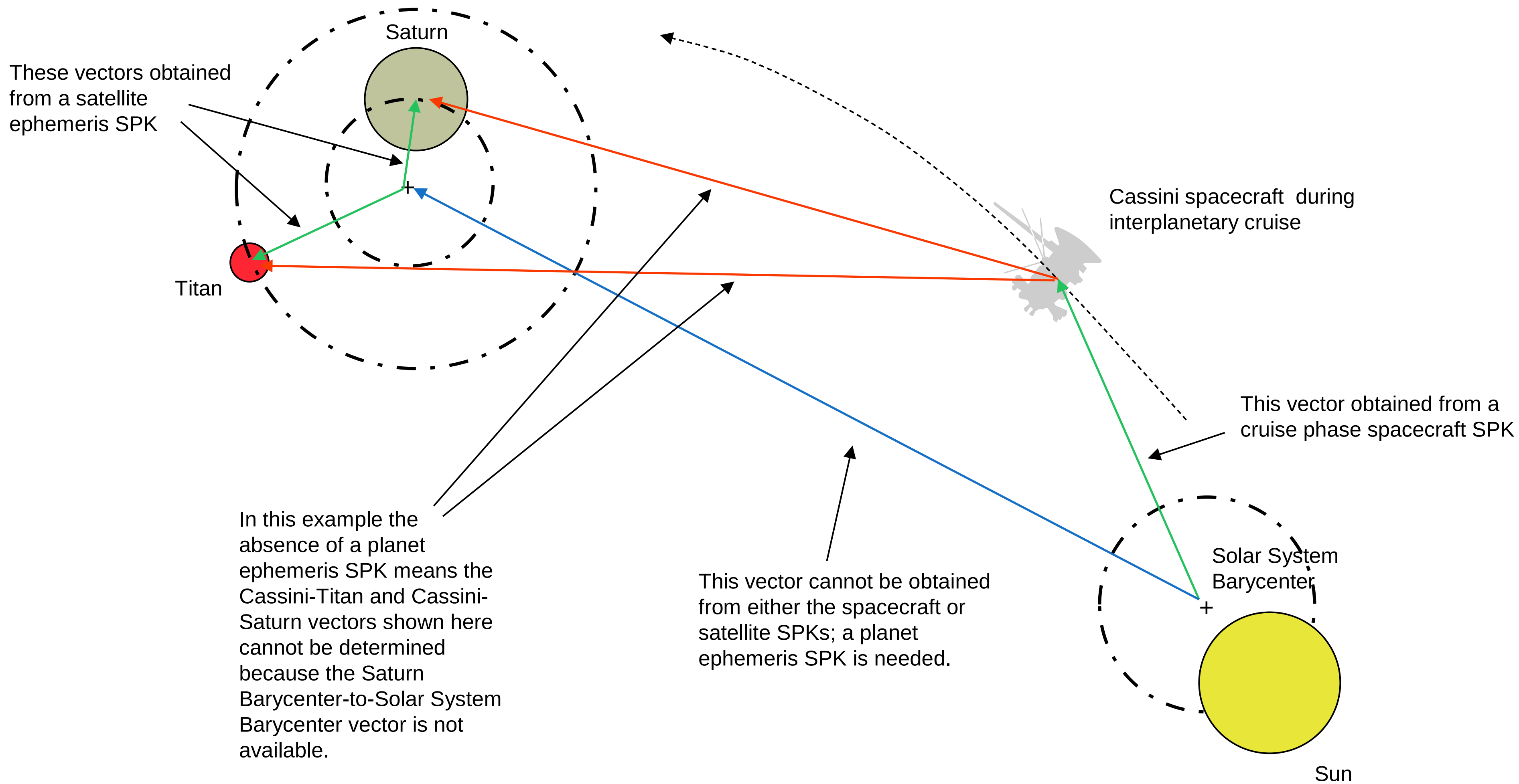
Problems Using SPK Files – 3a

- **An infrequent problem occurs when your SPK file(s) contain data for both target and observer, and cover the period of interest, but ephemeris data for an intermediate body needed to chain the target and observer together is missing.**
  - **Example: You load a spacecraft SPK containing ephemeris for Cassini (-82) relative to the solar system barycenter (0), and you load a satellite SPK containing the ephemeris for Titan (606) and Saturn (699) relative to the Saturn barycenter (6). But you forgot to load a planet SPK file that contains data for the Saturn barycenter**

relative to the solar system barycenter. The SPK software cannot "connect" Cassini to Titan or to Saturn. (See the drawing on the next page.)

– In this case, knowing what is the "Center Body" of movement for each target body is important; this is shown in SPACIT, and also in BRIEF summaries when the -c command line option is used.

Problems Using SPK Files - 3b (drawing)



Given…

available in SPK files not available in SPK files

Therefore…   can't be computed

Problems Using SPK Files - 4

- You see an error message to the effect that pole RA (right ascension) data cannot be found
  - You are requesting results in a body-fixed frame, but you have not loaded a SPICE PCK file that defines this frame.

Problems Using SPK Files - 5

Navigation and Ancillary Information Facility

- **Segment Masking: You've loaded sufficient data to chain together the target and observer, but the SPK subsystem can't make the connection.**
  - This can happen when a high-priority segment that can't be connected to both target and observer "masks" a lower-priority segment that can be connected.
  - Example: you want the state of earth as seen from the Galileo orbiter at a specified ephemeris time ET1.

» You have loaded SPK files providing:

- the state of the Galileo orbiter relative to the asteroid Gaspra
- the state of the orbiter relative to the sun
- the state of the earth relative to the earth-moon barycenter
- the states of the sun and earth-moon barycenter relative to the solar system barycenter

- » If an SPK segment for the orbiter relative to Gaspra covering ET1 has higher priority than the segment for the orbiter relative to the sun covering ET1, no connection between the orbiter and the earth will be made.
- » Solution:
  - Load an SPK file providing the ephemeris of Gaspra relative to the sun or the solar system barycenter (for a time interval containing ET1)

Problems Using SPK Files - 6

- **Other missing data… not obvious.**
  - You may need CK (and SCLK), FK or PCK data to construct your requested output frame.
- **Mistaking ET for UTC, or vice-versa.**

- **You must have loaded sufficient SPKs to be able to chain states to the solar system barycenter if doing aberration corrections.**
- **Using light time corrections requires target ephemeris data at the light time-corrected epoch.**
  - **If you're working near the beginning of an SPK file, the light time-corrected epoch may occur earlier than available data.**

Problems Using SPK Files - 7

Navigation and Ancillary Information Facility

- **You've assumed that:**

negative sign

state (observer, target) = - state (target, observer)

- **This is NOT true unless you have requested geometric states in both cases (i.e. no light time or stellar aberration corrections are applied)**

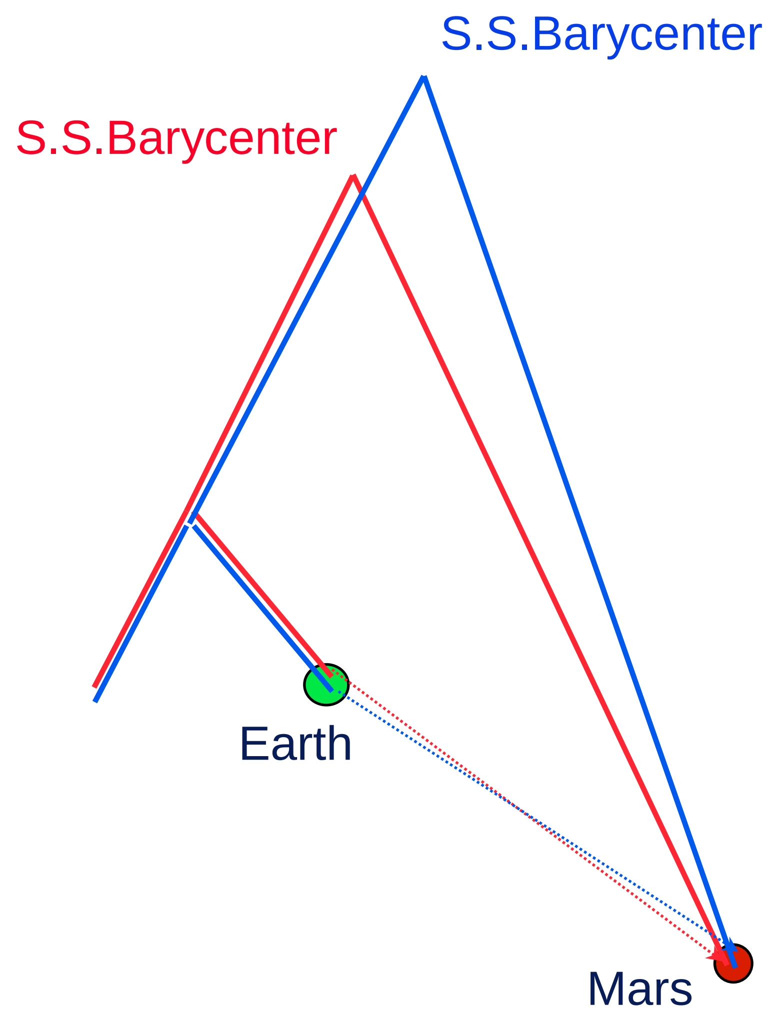
Don’t Mix Planet Ephemerides-1

- **With each new version of the JPL planetary ephemeris, the solar system barycenter moves with respect to the planets**

- **Changes in relative planet positions are much smaller than changes in the planet locations relative to the solar system barycenter**

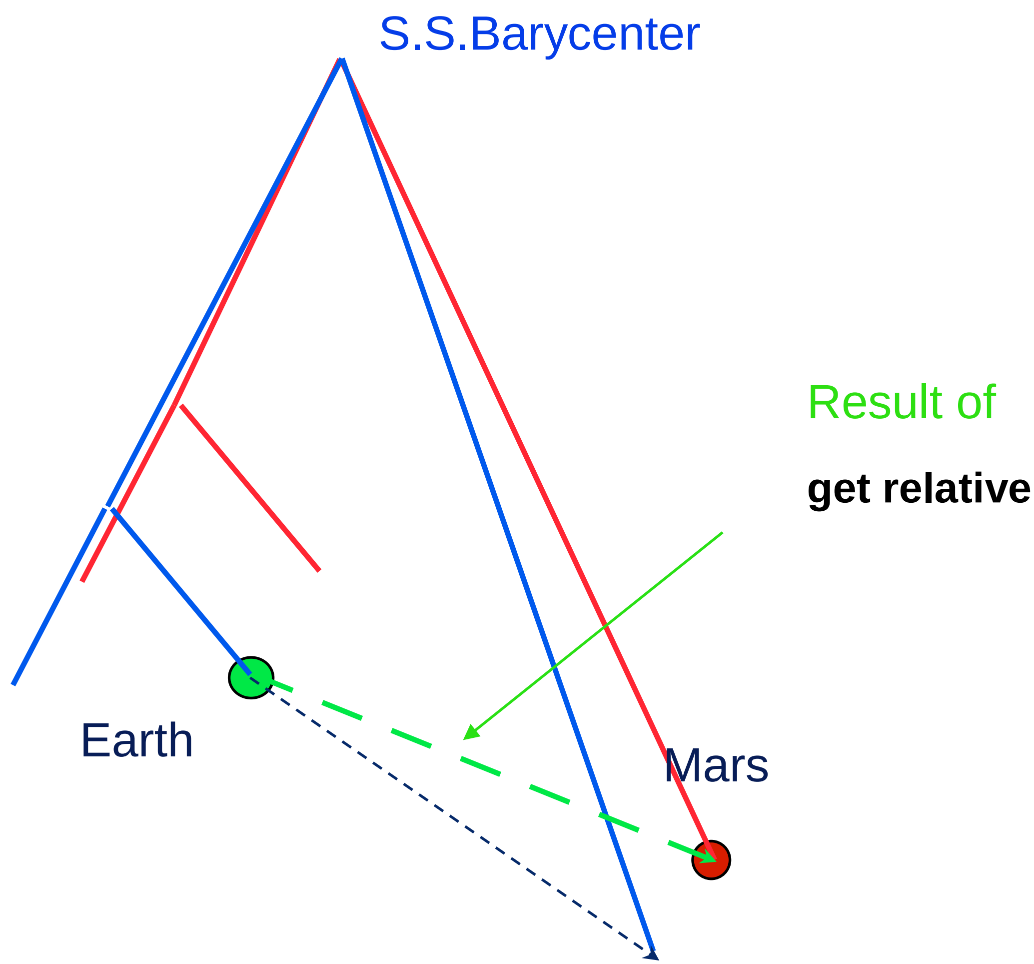
Don't Mix Planet Ephemerides-

Navigation and Ancillary Information Facility



2

- **SPICE allows you to “load” different planetary ephemerides (or portions of them)**
  - » **Potentially can subtract the solar system barycenter-relative positions from different** mixing **ephemerides to** **states** ephemerides
- **Don’t mix planetary ephemerides**
- **For JPL flight projects, a consistent set of ephemerides is provided to the mission team**



Effect of Aberration Corrections - 1

- **Angular offsets between corrected and uncorrected position vectors over the time span 2004 Jan 1 to 2005 Jan1**
  - **Mars as seen from MEX:**
    - » **LT+S vs NONE: .0002 to .0008 degrees**
    - » **LT vs NONE: .0006 to .0047 degrees**
  - **Earth as seen from MEX:**
    - » **LT+S vs NONE: .0035 to .0106 degrees**
    - » **LT vs NONE: .0000 to .0057 degrees**
  - **MEX as seen from Earth:**
    - » **LT+S vs NONE: .0035 to .0104 degrees**
    - » **LT vs NONE: .0033 to .0048 degrees**
  - **Sun as seen from Mars:**
    - » **LT+S vs NONE: .0042 to .0047 degrees**
    - » **LT vs NONE: .0000 to .0000 degrees**

Effect of Aberration Corrections - 2

- **Angular offsets between corrected and uncorrected position vectors over the time span 2004 Jan 1 to 2008 Jan1**
  – **Saturn as seen from CASSINI:**

» LT+S vs NONE: .0000 to .0058 degrees » LT vs
NONE: .0001 to .0019 degrees – Titan as seen from
CASSINI:
» LT+S vs NONE: .0000 to .0057 degrees » LT vs
NONE: .0000 to .0030 degrees – Earth as seen from
CASSINI:
» LT+S vs NONE: .0000 to .0107 degrees » LT vs
NONE: .0000 to .0058 degrees – CASSINI as seen from
Earth:
» LT+S vs NONE: .0000 to .0107 degrees » LT vs
NONE: .0000 to .0059 degrees – Sun as seen from
CASSINI:
» LT+S vs NONE: .0000 to .0059 degrees

» LT vs NONE: .0000 to .0000 degrees

Examples of Retrieving State Vectors – 1

- Example: find the geometric state of the MGS orbiter relative to Mars at the observation epoch ET, expressed in the J2000 reference frame.
  - CALL SPKEZR ('MGS', ET, 'J2000', 'NONE', 'MARS', STATE, LT)
  - The SPK subsystem locates an SPK segment containing the ephemeris of the orbiter relative to Mars covering epoch ET, interpolates the ephemeris data at ET, and returns the interpolated state vector.
- Example: find the geometric state of Titan relative to the earth at epoch ET, expressed in the J2000 reference frame.
  - CALL SPKEZR ('TITAN', ET, 'J2000', 'NONE', 'EARTH', STATE, LT) – The SPK subsystem looks up and interpolates ephemeris data to yield: » The state of the earth relative to the earth-moon barycenter (A)
    - » The state of the earth-moon barycenter relative to the solar system barycenter (B)
      - » The state of Titan relative to the Saturn system barycenter at ET (C)

» The state of the Saturn system barycenter relative to the solar system barycenter at ET (D)

– SPKEZR then returns the state vector » C + D – (A + B)

Examples of Retrieving State Vectors – 2

- **Example: find the apparent state of the Cassini orbiter relative to the DSN station DSS-14, expressed in the topocentric reference frame centered at DSS-14, at a specified observation epoch ET.**
  - CALL SPKEZR ('CASSINI', ET, 'DSS-14_TOPO', 'LT+S', 'DSS-14', STATE, LT)
  - The SPK subsystem looks up and interpolates ephemeris data to yield:
    - » The state of DSS-14 relative to the earth in the ITRF93 terrestrial reference frame (A)
    - » The state at ET of the earth relative to the earth-moon barycenter in the J2000 reference frame (B)
    - » The state at ET of the earth-moon barycenter relative to the solar system barycenter in the J2000 frame (C)

» The state at the light time-corrected epoch ET-LT of the Cassini orbiter relative to the Saturn system barycenter (other centers are possible) in the J2000 frame (D)

continued on next page

Examples of Retrieving State Vectors – 3

  - » The state at ET-LT of the Saturn system barycenter relative to the solar system barycenter in the J2000 frame (E)
- – The SPK subsystem also looks up transformation matrices to map states:
  - » From the J2000 frame to the ITRF93 terrestrial (earth bodyfixed) frame at the observation epoch ET (T1)
  - » From the ITRF93 terrestrial frame to the DSS-14-centered topocentric frame (T2)
- – SPKEZR then computes the J2000-relative, light-time corrected observer-target state vector

» E + D – ($(T1)^{-1}$ *A + B + C))

– SPKEZR corrects this vector for stellar aberration

» Call the result "V_J2000_apparent"

– and finally returns the requested state vector in the DSS-14 topocentric reference frame

» STATE = T2 * T1 * V_J2000_apparent

SPK File Structure

Navigation and Ancillary Information Facility

- A description of the SPK file structure is shown near the beginning of the “Making an SPK” tutorial.