UNIVERSITAT POLITÈCNICA DE CATALUNYA

STUDY OF CASSINI AND NEW HORIZONS TRAJECTORIES USING JPL SPICE LIBRARY

ANNEX

A thesis submitted by Roger Sala Marco for the BSc Degree in Aerospace Vehicle Engineering

June 10, 2019

Directed by:
Manel Soria Guerrero

Co-director:
Enrique García Melendo

Contents

Appendix A

SPICE functions

In this appendix, all the routines used from the SPICE library [2] are explained in more detail.

A.1 *cspice_ ckcov*

A.1.1 Abstract

This routine returns the coverage windows for a specified object in a specified CK file. The call is:

cov=cspice_ ckcov(ck, idcode, needav, level, tol, timsys, room)

A.1.2 Inputs and Outputs

The inputs are:

- **ck:** the name(s) of SPICE CK files.
- **idcode:** the CK ID code of an object, normally a spacecraft structure or instrument, for which pointing data are expected to exist in the specified CK file.
- **needav:** a flag indicating whether to consider segments having angular velocity when determining coverage. When *needav* is true, segments without angular velocity do not contribute to the coverage window; when *needav* is false, all segments for *idcode* may contribute to the coverage window. In this project this value is always '0'.
- **level:** the string defining the level (granularity) at which the coverage is examined. Allowed values and corresponding meanings are:
  - **'SEGMENT':** The output coverage window contains intervals defined by the start and stop times of segments for the object designated by *idcode*.

  - **'INTERVAL':** The output coverage window contains interpolation intervals of segments for the object designated by *idcode*. For type 1 segments, which do not have interpolation intervals, each epoch associated with a pointing instance is treated as a singleton interval; these intervals are added to the coverage window.

    All interpolation intervals are considered to lie within the segment bounds for the purpose of this summary: if an interpolation interval extends beyond the segment coverage interval, only its intersection with the segment coverage interval is considered to contribute to the total coverage.

- **tol:** the tolerance value expressed in ticks of the spacecraft clock associated with *idcode*. Before each interval is inserted into the coverage window, the interval is intersected with the segment coverage interval, and if the intersection is non-empty, it is expanded by *tol*: the left endpoint of the intersection interval is reduced by and the right endpoint is increase *tol*. Adjusted interval *tol* endpoints, when expressed as encoded SCLK, never are less than zero ticks. Any intervals that overlap as a result of the expansion are merged. The *tol* value in this project is set at '0'.

- **timsys:** the name of the time system to use in the output coverage window. *timsys* may have the values:

  - **'SCLK':** Elements of *cov* are expressed in encoded SCLK ("ticks"), where the clock is associated with the object designated by *idcode*.
  - **'TDB':** Elements of *cov* are expressed as seconds past J2000 TDB.

- **room:** the number of intervals for use as a workspace by the routine. This value should equal at least the number of intervals corresponding to *idcode* in *ck*.

The outputs are:

- **cov:** window containing the coverage for *idcode* available from *ck*. *cov* returns an empty set if *ck* lacks coverage for *idcode*.

  When *level* is *'INTERVAL'*, this is the set of time intervals for which data for *idcode* are present in the file *ck*. The array *cov* contains the pairs of endpoints of these intervals.

  When *level* is *'SEGMENT'*, *cov* is computed in a manner similar to that described above, but the coverage intervals used in the computation are those of segments rather than interpolation intervals within segments.

A.2 *cspice_spkcov*

A.2.1 Abstract

This function returns the coverage windows for a specified ephemeris object in a specified SPK file. The call is:

cover=cspice_spkcov(spk,idcode,room)

A.2.2 Inputs and Outputs

The inputs are:

- **spk:** the name, or cell of names, of SPICE SPK file(s).
- **idcode:** the SPK ID code of an object for which ephemeris data are expected to exist in the specified SPK file.
- **room:** the number of intervals for use as a workspace by the routine. This value should equal at least the number of intervals corresponding to *idcode* in *spk*.

The outputs are:

- **cover:** the window containing the coverage for *idcode*, i.e. the set of time intervals for which *idcode* data exist in the file *spk*. The array *cover* contains the pairs of endpoints of these intervals.

  The interval endpoints contained in *cover* are ephemeris times, expressed as seconds past J2000 TDB.

  cover returns an empty set if *spk* lacks coverage for *idcode*.

A.3 *cspice_furnsh*

A.3.1 Abstract

This routine loads SPICE kernel files into MATLAB. The call is:

cspice_furnsh(file)

A.3.2 Inputs and Outputs

The inputs are:

- **file:** name of a SPICE kernel file. The file may be either binary or text. If the file is a binary SPICE kernel it will be loaded into the appropriate SPICE subsystem. If *file* is a SPICE text kernel it will be loaded into the kernel pool. If *file* is a SPICE meta-kernel containing initialization instructions (through use of the correct kernel pool variables), the files specified in those variables will be loaded into the appropriate SPICE subsystem.

There are no outputs.

A.4 *cspice_str2et*

A.4.1 Abstract

This function converts a string representing an epoch to a double precision value representing the number of TDB seconds past the J2000 epoch corresponding to the input epoch. The call is:

et=cspice_str2et(str)

A.4.2 Inputs and Outputs

The inputs are:

- **str:** Any scalar or array of strings recognized by SPICE as an epoch.

The outputs are:

- **et:** The scalar or N-vector of double precision number of TDB seconds past the J2000 epoch that corresponds to the input *str*.

A.5 *cspice_ckobj*

A.5.1 Abstract

This routine returns the set of ID codes of all objects in a specified CK file. The call is:

ids=cspice_ckobj(ck,room)

A.5.2 Inputs and Outputs

The inputs are:

- **ck:** the name(s) for SPICE CKs.
- **room:** the maximum number of CK IDs to return from *ck*.

The outputs are:

- **ids:** the set of unique CK ID codes for which pointing data exists in *ck*.

A.6 *cpice_spkobj*

A.6.1 Abstract

This function returns the set of ID codes of all objects in a specified SPK file. The call is:

ids=cspice_spkobj(spk,room)

A.6.2 Inputs and Outputs

The inputs are:

- **spk:** the name, or cell of names, of SPICE SPK file(s).
- **room:** an integer scalar defining the maximum number of SPK IDs to return from *spk*.

The outputs are:

- **ids:** an array containing the set of unique NAIF ID codes for which ephemeris data exists in *spk*.

A.7 *cspice_kclear*

A.7.1 Abstract

This function clears the KEEPER system: unload all kernels, clears the kernel pool, and re-initialize the system. The call is:

cspice_kclear

A.8 *cspice_bodvrd*

A.8.1 Abstract

This function returns from the kernel pool the double precision values of an item associated with a body. The call is:

values=cspice_bodvrd(body, item, mxn)

A.8.2 Inputs and Outputs

The inputs are:

- **body:** The scalar string name of the body for which the data *item* is requested. The string lacks case sensitivity.
- **item:** The scalar string naming the property of 'body' to return, e.g. RADII, GM, POLE_RA. The string is case-sensitive.
- **mxn:** The scalar integer maximum number of values to return, this value must equal or exceed the size of requested kernel variable *item*.

The outputs are:

- **values:** An array of the double precision values associated with the variable.

A.9 *cspice_ spkezr*

A.9.1 Abstract

This function returns the state (position and velocity) of a target body relative to an observing body, optionally corrected for light time (planetary aberration) and stellar aberration[1]. The call is:

[state, lt] = cspice_ spkezr(targ, et, ref, abcorr, obs)

A.9.2 Inputs and Outputs

The inputs are:

- **targ:** name of a target body. Optionally, you may supply the integer ID code for the object as an integer string, i.e. both 'MOON' and '301' are legitimate strings that indicate the Moon is the target body. The target and observer define a state vector whose position component points from the observer to the target.
- **et:** the ephemeris time(s), expressed as seconds past J2000 TDB, at which the state of the target body relative to the observer is to be computed. *et* refers to time at the observer's location.
- **ref:** the name of the reference frame relative to which the output state vector should be expressed. This may be any frame supported by the SPICE system, including built-in frames (documented in the Frames Required Reading) and frames defined by a loaded frame kernel (FK).

  When *ref* designates a non-inertial frame, the orientation of the frame is evaluated at an epoch dependent on the selected aberration correction.
- **abcorr:** aberration corrections to apply to the state of the target body to account for one-way light time and stellar aberration (see Appendix **??**).
- **obs:** name of a observing body. Optionally, you may supply the integer ID code for the object as an integer string, i.e. both 'MOON' and '301' are legitimate strings that indicate the Moon is the observing body.

The outputs are:

- **state:** the Cartesian state vector(s) representing the position and velocity of the target body relative to the specified observer. *state* is corrected for the specified aberrations, and is expressed with respect to the reference frame specified by *ref*. The first three components of *state* represent the x-, y- and z-components of the position of the target; the last three components form the corresponding velocity vector.

[1] In this project no aberration correction has been applied.

The position component of *state* points from the observer's location at *et* to the aberration-corrected location of the target. The velocity component of *state* is the derivative with respect to time of the position component of *state*. Units are always [km] and [km/s].

- **lt:** the value(s) of the one-way light time between the observer and target in seconds. If the target state is corrected for aberrations, then *lt* is the one-way light time between the observer and the light time corrected target location.

A.10 *cspice_occult*

A.10.1 Abstract

This function determines the occultation condition (not occulted, partially, etc.) of one target relative to another target as seen by an observer at a given time.

The surfaces of the target bodies may be represented by triaxial ellipsoids or by topographic data provided by DSK files[2]. The call is:

occult_code = cspice_occult (target1, shape1, frame1,...
target2, shape2, frame2,...
abcorr, observer, time)

A.10.2 Inputs and Outputs

The inputs are:

- **target1:** is the name of the first target body. Both object names and NAIF IDs are accepted, e.g. both 'Moon' and '301' are accepted.
- **shape1:** is a string indicating the geometric model used to represent the shape of the first target body. The supported options are 'ELLIPSOID', 'DSK/UNPRIORITIZED' and 'POINT', but only the 'ELLIPSOID' model is used.
- **frame1:** is the name of the body-fixed, body-centered reference frame associated with the first target body, e.g. 'IAU_SATURN' (for Saturn) and 'ITRF93' (for the Earth).
- **target2:** is the name of the second target body. Both object names and NAIF IDs are accepted, e.g. both 'Moon' and '301' are accepted.
- **shape2:** is a string indicating the geometric model used to represent the shape of the second target body. The supported options are the same as the ones on shape1 but, as said before, the 'ELLIPSOID' model is used.

[2]In this project all bodies are represented by triaxial ellipsoids.

- **frame2:** is the name of the body-fixed, body-centered reference frame associated with the second target body, e.g. 'IAU_SATURN' (for Saturn) and 'ITRF93' (for the Earth).
- **abcorr:** indicates the aberration corrections to be applied to the state of each target body to account for one-way light time (see Appendix **??**). Stellar aberration corrections are ignored since these corrections do not improve the accuracy of the occultation determination.
- **observer:** is the name of the body from which the occultation is observed. See the description of *target1* for more details.
- **time:** is the observation time in seconds past the J2000 epoch.

The outputs are:

- **occult_code:** is an integer occultation code indicating the geometric relationship of the three bodies.

  Possible *occult_code* values and meanings are given on the table below.

ID code	Meaning
-3	Total occultation of first target by second
-2	Annular occultation of first target by second
-1	Partial occultation of first target by second
0	No occultation
1	Partial occultation of second target by first
2	Annular occultation of second target by first
3	Total occultation of second target by first

Table A.1: ID codes for occultations.

A.11 *cspice_et2utc*

A.11.1 Abstract

This function converts an input time from ephemeris seconds past J2000 to Calendar, Day-of-Year, or Julian Date format, UTC. The call is:

utcstr = cspice_et2utc(et, format, prec)

A.11.2 Inputs and Outputs

The inputs are:

- **et:** the ephemeris time(s) expressed as ephemeris seconds past J2000.
- **format:** the format flag describing the output time string, it may be any of the following:

- **'C':** Calendar format, UTC
- **'D':** Day-of-Year format, UTC
- **'J':** Julian Date format, UTC
- **'ISOC':** ISO Calendar format, UTC
- **'ISOD':** ISO Day-of-Year format, UTC

- **prec:** number of decimal places of precision to which fractional seconds (for Calendar and Day-of-Year formats) or days (for Julian Date format) are to be computed.

The outputs are:

- **utcstr:** the array of time string(s) equivalent to the input epoch 'et', in the specified 'format'.

A.12 *cspice_bodn2c*

A.12.1 Abstract

This function translates the name of a body or object to the corresponding SPICE integer ID code. The call is:

[code, found]=cspice_bodn2c(name)

A.12.2 Inputs and Outputs

The inputs are:

- **name:** name(s) of a body or object, such as a planet, satellite, comet, asteroid, barycenter, DSN station, spacecraft, or instrument, "known" to the SPICE system, whether through hard-coded registration or run-time registration in the SPICE kernel pool.

The outputs are:

- **code:** containing the SPICE code(s) assigned either by SPICE or the user to *name*.
- **found:** flag(s) indicating if the kernel subsystem translated *name* to a corresponding *code*.

A.13 *cspice_getfov*

A.13.1 Abstract

This function returns the field-of-view parameters for a user specified instrument. The call is:

[shape,frame,bsight,bounds]=cspice_getfov(instid,room)

A.13.2 Inputs and Outputs

The inputs are:

- **instid:** The integer NAIF ID for the instrument of interest.
- **room:** The max number of double precision *bounds* vectors to return.

The outputs are:

- **shape:** A string describing the FOV shape for instrument *instid*. Possible values:
  - "POLYGON"
  - "RECTANGLE"
  - "CIRCLE"
  - "ELLIPSE"
- **frame:** A string identifying the frame in which the FOV is defined.
- **bsight:** A double precision 3-vector pointing in the direction of the FOV center (boresight).
- **bounds:** An array of 3-vectors pointing to the "corners" of the instrument FOV.

A.14 *cspice_sincpt*

A.14.1 Abstract

This function computes the surface intercept of the ray on a target body at a specified epoch, optionally corrected for light time and stellar aberration, given an observer and a direction vector defining a ray. The call is:

[spoint, trgepc, srfvec, found]=cspice_sincpt(method, target, et, fixref, abcorr, obsrvr, dref, dvec)

A.14.2 Inputs and Outputs

The inputs are:

- **method:** A scalar string providing parameters defining the computation method to be used. The choices currently supported are 'ELLIPSOID' and 'DSK', but only 'ELLIPSOID' is used. The string is case-insensitive.
- **target:** The scalar string name of the target body. The string is case-insensitive.

- **et:** The double precision scalar epoch of participation of the observer, expressed as ephemeris seconds past J2000 TDB: *et* is the epoch at which the observer's state is computed.
- **fixref:** The scalar string naming the body-fixed, body-centered reference frame associated with the target body.
- **abcorr:** The scalar string aberration correction to be applied when computing the observer-target state and the orientation of the target body (see Appendix **??**).
- **obsrvr:** The scalar string name of the observing body. The string is case-insensitive.
- **dref:** The name of the reference frame relative to which the ray's direction vector is expressed.
- **dvec:** The double precision 3-vector defining the pointing vector emanating from the observer, specified relative to the reference frame designated by *dref*. The intercept with the target body's surface of the ray defined by the observer and *dvec* is sought.

The outputs are:

- **spoint:** Double precision 3-vector defining surface intercept point on the target body of the ray defined by the observer and the direction vector. It is expressed in Cartesian coordinates, relative to the target body-fixed frame designated by *fixref*. The components of *spoint* are given in units of '[km].
- **trgepc:** The scalar double precision "intercept epoch." This is the epoch at which the ray defined by *obsrvr* and *dvec* intercepts the target surface at *spoint*. It is expressed as seconds past J2000 TDB.
- **srfvec:** A double precision 3-vector defining the position vector from the observer at *et* to *spoint*. It is expressed in the target body-fixed reference frame designated by *fixref*, evaluated at *trgepc*. The components of *srfvec* are given in units of [km].
- **found:** A scalar logical indicating whether or not the ray intersects the target. If an intersection exists, *found* will return as true. If the ray misses the target, *found* will return as false.

A.15 *cspice_ilumin*

A.15.1 Abstract

This function computes the illumination angles (phase, solar incidence, and emission) at a specified surface point of a target body. The call is:

[trgepc, srfvec, phase, solar, emissn] = cspice_ilumin(method, target, et, fixref, abcorr, obsrvr, spoint)

A.15.2 Inputs and Outputs

The inputs are:

- **method:** A scalar string providing parameters defining the computation method to be used. The only choice currently supported is 'ELLIPSOID'. The string is case-insensitive.
- **target:** The scalar string name of the target body. The string is case-insensitive.
- **et:** The scalar double precision epoch, specified in ephemeris seconds past J2000, at which the apparent illumination angles at the specified surface point on the target body, as seen from the observing body, are to be computed.
- **fixref:** The scalar string naming the body-fixed, body-centered reference frame associated with the target body.
- **abcorr:** The scalar string aberration correction to use in computing the location of the surface point, the orientation of the target body, and the location of the Sun (see Appendix **??**).
- **obsrvr:** The scalar string name of the observing body. The string is case-insensitive.
- **spoint:** A double precision 3-vector defining the surface point on the target body, expressed in Cartesian coordinates, relative to the body-fixed target frame designated by *fixref*. The components of *spoint* have units of [km].

The outputs are:

- **trgepc:** It is the "surface point epoch" and it is expressed as seconds past J2000 TDB.
- **srfvec:** A double precision 3-vector defining the position vector from the observer at *et* to *spoint*. It is expressed in the target body-fixed reference frame designated by *fixref*, evaluated at *trgepc*. The components of *srfvec* are given in units of [km].
- **phase:** The scalar double precision defining the angle between the *spoint-obsrvr* vector and the *spoint-sun* vector. Units are radians. The range of *phase* is [0, pi].
- **solar:** This is the angle between the surface normal vector at *spoint* and the *spoint-sun* vector. Units are radians. The range of *solar* is [0, pi].
- **emissn:** The scalar double precision defining the angle between the surface normal vector at *spoint* and the *spoint-obsrvr* vector. Units are radians. The range of *emissn* is [0, pi].

A.16 *cspice_pxform*

A.16.1 Abstract

This function returns the matrix that transforms position vectors from one specified frame to another at a specified epoch. The call is:

rotate=cspice_pxform(from,to,et)

A.16.2 Inputs and Outputs

The inputs are:

- **from:** the name of a reference frame in which a position is known.
- **to:** the name of a reference frame in which it is desired to represent the position.
- **et:** epoch(s) in ephemeris seconds past the epoch of J2000 (TDB) at which the to evaluate the position transformation operator(s).

The outputs are:

- **rotate:** operator(s) that transform position vector(s) from the reference frame *from* to frame *to* at epoch *et*.

A.17 *cspice_surfpt*

A.17.1 Abstract

This routine determines the intersection of a line-of-sight vector with the surface of an ellipsoid. The call is:

[point,found]=cspice_surfpt(positn,u,a,b,c)

A.17.2 Inputs and Outputs

The inputs are:

- **positn:** the position of an observer with respect to the center of an ellipsoid expressed in the body fixed coordinates of the ellipsoid.
- **u:** the direction vector emanating from *positn*.
- **a:** length in kilometers of the semi-axis of the ellipsoid parallel to the x-axis of the body-fixed reference frame.
- **b:** length in kilometers of the semi-axis of the ellipsoid parallel to the y-axis of the body-fixed reference frame.

- **c:** length in kilometers of the semi-axis of the ellipsoid parallel to the z-axis of the body-fixed reference frame.

The outputs are:

- **point:** the location on the ellipsoid at which the u intercepts the ellipsoid if the interception exists, *point* returns (0.d, 0.d, 0.d) if u does not intersect the ellipsoid.
- **found:** a flag indicating whether the intersection between the ellipse and u exists (TRUE) or not (FALSE).

A.18 *cspice_reclat*

A.18.1 Abstract

This function converts rectangular (Cartesian) coordinates to latitudinal coordinates. All coordinates are expressed as double precision values. The call is:

[radius,lon,lat]=cspice_reclat(rectan)

A.18.2 Inputs and Outputs

The inputs are:

- **rectan:** the array(s) containing the rectangular coordinates of the position or set of positions. Units are [km].

The outputs are:

- **radius:** the value(s) describing the distance of the position from the origin. The argument *radius* returns in the same units associated with *rectan*.
- **lon:** the value(s) describing the angle of the position from the XZ plane measured in radians.
- **lat:** the value(s) describing the angle of the position from the XY plane measured in radians.

A.19 *cspice_latsrf*

A.19.1 Abstract

This function maps an array of planetocentric longitude/latitude coordinate pairs to surface points on a specified target body.

The surface of the target body may be represented by a triaxial ellipsoid or by topographic data provided by DSK files. The call is:

srfpts=cspice_latsrf(method, target, et, fixref, lonlat)

A.19.2 Inputs and Outputs

The inputs are:

- **method:** is a short string providing parameters defining the computation method to be used. In the syntax descriptions below, items delimited by brackets are optional.

  The different methods can be:

  - **'ELLIPSOID':** The surface point computation uses a triaxial ellipsoid to model the surface of the target body. The radii of the ellipsoid must be available in the kernel pool.
  - **'DSK/UNPRIORITIZED':** The surface point computation uses topographic data to model the surface of the target body. These data must be provided by loaded DSK files.

- **target:** is the name of the target body.

- **et:** is the epoch for which target surface data will be selected, if the surface is modeled using DSK data. In this case, only segments having time coverage that includes the epoch *et* will be used.

  et is ignored if the target is modeled as an ellipsoid.

  et is expressed as TDB seconds past J2000 TDB.

- **fixref:** is the name of a body-fixed reference frame centered on the target body. *fixref* may be any such frame supported by the SPICE system, including built-in frames (documented in the Frames Required Reading) and frames defined by a loaded frame kernel (FK).

- **lonlat:** is an array of pairs of planetocentric longitudes and latitudes of surface points.

The outputs are:

- **srfpts:** is an array of target body surface points corresponding to the pairs of coordinates in the input *lonlat* array.

Appendix B

Aberration corrections

There are many types of aberration corrections that can be applied on the SPICE functions [1]. All of them are explained below:

- **'NONE':** Apply no correction. Return the geometric state of the target body relative to the observer.

The following values of *abcorr* apply to the "reception" case in which photons depart from the location of the target at the light-time corrected epoch et-lt and *arrive* at the location of the observer at *et*:

- **'LT':** Correct for one-way light time (also called "planetary aberration") using a Newtonian formulation. This correction yields the state of the target at the moment it emitted photons arriving at the observer at *et*.

  The light time correction uses an iterative solution of the light time equation (see Particulars for details). The solution invoked by the **"LT"** option uses one iteration.

- **'LT+S':** Correct for one-way light time and stellar aberration using a Newtonian formulation. This option modifies the state obtained with the **"LT"** option to account for the observer's velocity relative to the solar system barycenter. The result is the apparent state of the target—the position and velocity of the target as seen by the observer.

- **'CN':** Converged Newtonian light time correction. In solving the light time equation, the **"CN"** correction iterates until the solution converges (three iterations on all supported platforms).

  The **"CN"** correction typically does not substantially improve accuracy because the errors made by ignoring relativistic effects may be larger than the improvement afforded by obtaining convergence of the light time solution. The **"CN"** correction computation also requires a significantly greater number of CPU cycles than does the one-iteration light time correction.

- **'CN+S':** Converged Newtonian light time and stellar aberration corrections.

The following values of *abcorr* apply to the "transmission" case in which photons *depart* from the observer's location at *et* and arrive at the location of the target at the light-time corrected epoch et+lt:

- **'XLT':** "Transmission" case: correct for one-way light time using a Newtonian formulation. This correction yields the state of the target at the moment it receives photons emitted from the the location of the observer at *et*.
- **'XLT+S':** "Transmission" case: correct for one-way light time and stellar aberration using a Newtonian formulation This option modifies the state obtained with the **"XLT"** option to account for the velocity of the observer relative to the solar system barycenter. The position component of the computed target state indicates the direction that photons emitted from the location of the observer must be "aimed" to hit the target.
- **'XCN':** "Transmission" case: converged Newtonian light time correction.
- **'XCN+S':** "Transmission" case: converged Newtonian light time and stellar aberration corrections.

Bibliography

[1] Function cspice_spkezr. https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/IDL/icy/cspice_spkezr.html. [Accessed: 2019-06-08].

[2] MICE Index. https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/MATLAB/mice/. [Accessed: 2019-06-08].