Initiation to the SPICE library

January 2021

Teresa Peña Mercadé

Degree in Aerospace Engineering

Universitat Politècnica de Catalunya, BarcelonaTech (UPC)
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Contents

1 Introduction

The purpose of this document is to provide useful information for those who use SPICE library for the first time.

Thus, first there is an explanation of the different types of Kernels that can be found, in order to know the content of each and be able to use them correctly. After, some MICE functions are exposed with an example to see their structure and their input data. The last two sections are related with common errors and webs that can be helpful to find different information.

It should be noted that this document is designed to be a backup of the class exercises, as well as a source to find information quickly.

2 Kernels

Depending on the data and information they contain, kernels can be classified in the next groups:

- CK (Camera matrix Kernels) [*.bc*]: They provide the orientation of an object or structure relative to a specified reference frame. To use them it is required an Spacecraft Clock Kernel (SCLK) and when they contain orientation of project-defined frames, also an FK. Sometimes three types of C-kernels are present: Imaging Science Subsystem (ISS) image navigation data, Attitude and Articulation Control Subsystem (AACS) reconstructed and predicted orientation data.
- EK (Events Kernels): Their purpose is to store science plans (ESP), instruments and spacecraft commands (ESQ), experiment notebooks and ground data system logs (ENB) and Imaging Science Subsystem (ISS) mechanical events that may be seen as interference in other science data.
- FK (Frames Kernels) [*.tf*]: They contain definitions and specifications of relationships between reference frames to define them.
- IK (Instrument Kernels) [*.ti*]: These kernels describe how an instrument is mounted to the instrument platform. They also offer specifications of the principal axes and the instrument, such as its field-of-view size, shape and orientation, as well as other possible useful instrument-specific information. Sometimes they need data from SPK and CK files to carry out some subroutines.
- LSK (Leapsecond Kernels) [*.tls*]: They contain a tabulation of leapseconds and the values of other constants required to perform a transformation between Universal Time Coordinated (UTC) and Ephemeris Time (ET). They are required for all SPICE computations.
- PCK (Planetary Constant Kernels) [*.tpc*]: They provide attitude, size and shape for planets, their satellites, the sun, comets and asteroids, as well as other similar constants such as parameters for gravitational, atmospheric or rings models.
- SCLK (Spacecraft Clock Kernels) [*.tsc*]: These kernels contain the information required to perform a mapping between Ephemeris Time (ET) and spacecraft on-board time (SCLK).
- SPK (Spacecraft Planet Kernels) [*.bsp*]: Their purpose is to allow ephemerides (positions) of any collection of solar system bodies, spaceraft or other objects in 3D space. Therefore, it is possible to obtain trajectories with their data. To use them a Leapsecond Kernel (LSK) is required and, in case of providing the position relative to a non-inertial and project-defined frame, FK and PCK are also necessary.

At the same time, each of the previous type of kernels can be encompassed inside a more general classification in:

- Generic kernels: They are independent of any particular flight project. Under this category SPK, PCK, LSK, FK can be found.
  - LSK: The latest version of the Leapsecond Kernel is the naif0012.tls (`https://naif.jpl.nasa.gov/pub/naif/generic_kernels/lsk/naif0012.tls`).

- SPK:
  - Planets: The de430.bps file (`https://naif.jpl.nasa.gov/pub/naif/generic_kernels/spk/planets/de430.bsp`) provides more accurate ephemeris for the Moon than de431.bps. Regarding de432s.bps, it has been developed expressly for the New Horizons project (updated ephemeris of the Pluto system barycenter).
  - Satellites: These kernels contain the position and velocity of a natural body satellite relative to the barycenter of the planet. The naming schema of these files is nnnxxx.bps, where nnn refers to the first three letters of the name of the planet and xxx is a unique number (without value to SPICE users). When there is more than one file for a planet, each file contains ephemeris data for a unique set of satellites, which means that it is necessary to check which one is adequate in every case. For example, if the satellite of analysis is Rhea, it is required to download the file sat427.bsp (`https://naif.jpl.nasa.gov/pub/naif/generic_kernels/spk/satellites/sat427.bsp`).

- Mission operations kernels: They contain data related to a specific mission. For example, the SPK files for Cassini provide the ephemerides of this spacecraft as well as for solar system bodies important to the mission.

  The files can be obtained from two different sources (although both are related): `https://naif.jpl.nasa.gov/pub/naif/` or `https://naif.jpl.nasa.gov/pub/naif/pds/data/`. In the first case, the next step is to select the mission, for example, CASSINI, and download the kernels needed. With the second option, the search of the mission is done through a different nomenclature (for CASSINI it is co-s_j_e_v-spice-6-v1.0/ and then cosp_1000/)). It should be noted that sometimes they display the same files and others, one of them is more extensive (normally the first option).
  - LSK: The file naif0012.tls can also be used with a mission (`https://naif.jpl.nasa.gov/pub/naif/CASSINI/kernels/lsk/naif0012.tls`).

3 MICE functions

A list of some MICE functions is provided, alongside its formulation and an example. The definitions are extracted from [3] and some of the examples are related between them. In Appendix A it can be found a brief description (without examples) of several other functions. It is worth knowing that many other functions can be found in the previous reference.

- **cspice_furnsh:** It loads SPICE kernels files into MATLAB. The input is the path of the file.

```
cspice_furnsh('/Volumes/Disc_extern/TFG/SPICE//kernels/naif0012.tls')
```

- **cspice_bodc2n:** It returns the body name corresponding to an input numeric ID value.

```
code=399;
[name,found]=cspice_bodc2n(code);
%Command Window results
name='EARTH'
found=1
```

- **cspice_bodn2c:** It is the inverse function to the previous. Therefore, it returns the integer ID code of a body.

```
body='SATURN';
[code,found]=cspice_bodc2n(body);
%Command Window results
code=699
found=1

instrument='CASSINI_ISS_NAC'; % Cassini Imaging Science System Narrow Angle Camera
[code,found]=cspice_bodc2n(instrument);
%Command Window results
code_ins=-82360
found=1
```

- **cspice_str2et:** It 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.

```
utctime='2000-01-01 T00:00:00';
et0=cspice_str2et (utctime);
%Command Window results
et0=-4.3136e+04
```

- **cspice_spkezr:** It returns the state (position in km and velocity in km/sec) of a target body relative to an observing body and the value of the one-way light time between the observer and the target in seconds. The required data is the target, the ephemeris time, the reference frame, the aberration correction and the observing body.

```
frame='ECLIPJ2000'; % Referece frames
abcorr='NONE'; % No corrections
observer='3'; % Earth System  barycenter
[dEarth,lt] = cspice_spkezr(name,et0,frame,abcorr,observer); % Earth state (km,km/s)
%Command Window results
dEarth=[3.8592e+03;2.9396e+03;-444.1733;-0.0068;-0.0097;2.0099e-5]
lt=0.0162
```

- **cspice_spkpos:** It is similar to the previous function, but in this case it only return the position of the target and not its velocity. The input data is the same.

- **cspice_spkobj:** It return the set of ID codes of all objects in a specified SPK file. It is necessary to introduce as data the spk file, the maximum number of IDs to return.

```
file='/Volumes/Disc_extern/TFG/SPICE//kernels/010420R_SCPSE_EP1_JP83.bsp';
ids=cspice_spkobj(file,100);
%Command Window results
ids=[-82;3;5;6;10;301...]
```

*IMPORTANT: The input variable (file) has to contain the path to the SPK file, otherwise the following message will appear: *Error using mice SPICE(FILENOTFOUND): [SPKOBJ→ GETFAT] The kernel file does not exist.*

- **cspice_spkcov:** It returns the coverage window for a specified ephemeris object in a specific SPK file. The given data for the function is the file, the ID code and the number of intervals for use as a workspace by the routine.

```
cover=cspice_spkcov(file, ids(1), 1000);
%Command Window results
cover=[-1.1700e+07;3.8664e+07]
```

- **cspice_timout:**It converts an input epoch represented in TDB seconds past the TDB epoch of J2000 to a character string formatted to the specifications of a user's format picture.

```
time=cspice_timout(cover(1), 'YYYY MON DD HR:MN:SC.### (TDB) ::TDB' );
%Command Window results
time='1999 AUG 19 02:00:00.000 (TDB)'
```

- **cspice_wnintd:** It inserts an interval into a double precision window. It is necessary to introduce the left and right endpoints of the interval and an optional input window containing zero or more intervals. The inclusions of this window argument, called *window_i*, results in an output window consisting of a union of the data in *window_i* and the window defined as [left,right].

```
window=cspice_wninsd(3,5);
%Command Window
window=[3;5]
%Representing the interval [3,5]

window=cspice_wninsd(1,6,window)
%Command Window results
window=[1;6]
%Representing the interval [1,6]

window=cspice_wninsd(7,9,window)
%Command Window results
window=[3;5;7;9]
%Representing the intervals [3,5] [7,9]

window=cspice_wninsd(6,8,window)
%Command Window results
window=[3;5;6;9]
%Representing the intervals [3,5] [6,9]
```

- **cspice_vsep:** It returns the separation angle in radians between two double precision, 3-vectors. The input data are two vectors.

```
angle=cspice_vsep(dEarth(1:3,1),dMoon(1:3,1)) % Where dMoon is the position of the Moon at et0 (first
    three raws of the vector) obtained in the same way as dEarth.
%Command Window results
angle=3.1416
```

- **cspice_fovtrg:** It determines if a specified ephemeris object is within the field-of-view (FOV) of a specified instrument at a given time. The input arguments are the name of the instrument, the target name, shape and frame, the aberration correction, the observer and the ephemeris time. The output (*visibl*) is true if the target is fully or partially in the field-of-view of the instrument at the indicated time.

```
instrument='CASSINI_ISS_NAC';
target='RHEA';
target_shape='ELLIPSOID';
target_frame='IAU_RHEA';
abcorr='XLT+S';
observer='CASSINI';
visib1 = cspice_fovtrg(instrument,target,target_shape,target_frame,abcorr,observer,et)
```

* ATTENTION: The previous example requires specific kernels of the Cassini mission.

- **cspice_bodvrd:** It determines the value of an item associated to a body. The inputs are the body, the item and the maximum number of values to return.

```
body='JUPITER';
item='RADII'; % It must be in capital letters
max=3;
radii=cspice_bodvrd(body,item,max);
%Command Window results
radii=[71492;71492;66854] % Where the first component is the equatorial radius and the third, the polar
    one.
```

*IMPORTANT: It is needed to use the correct kernels to obtain the output values.

- **cspice_gipool:** It returns the integers values associated to a variable (*name*) beginning at the indicated index (*start*). The input arguments are the name of a pool variable, the value for the index indicating the first component of the data vector assigned to *name* for return (1 for all elements) and the maximum number of components to return (*room*).

```
name=INS-82360_WAVELENGTH_RANGE;
start=1;
room=2;
[values,f]=cspice_gipool(name,start,room);
%Command Window results
values=[200;1100]
f=1

name=INS-82360_WAVELENGTH_RANGE;
start=2;
room=2;
[values,f]=cspice_gipool(name,start,room);
%Command Window results
values=1100
f=1
```

* ATTENTION: This example is used for an specific code (*Navigation*), related to the Cassini Orbiter. The values of the different pool variables can be found in [1].

- **cspice_gdpool:** It has the same structure and purpose as the previous function, but in this case it returns double precision values.

```
name=INS-82360_FOCAL_LENGTH;
start=1;
room=1;
[values,f]=cspice_gdpool(name,start,room);
%Command Window results
values=2.0034e+03
f=1
```

- **cspice_getfov:** It returns the field-of-view (FOV) parameters for an specific instrument. The inputs, therefore, are the ID code for the instrument and the maximum number of double precision vectors to return (*room*). Among the outputs, one can find the FOV shape, the name of the frame in which the FOV is defined, the boresight* and the bounds **

```
instid=code_ins % Extracted from the example of the function cspice\_bodn2c
room=10;
[shape, iframe, boresight, bounds] = cspice_getfov(instid, room);
%Command Window results
shape='RECTANGLE';
iframe='CASSINI_ISS_NAC'
```

```
boresight=[0;0;1]
bounds=[0.0031  -0.0031  -0.0031  0.0031;  0.0031  0.0031...]
```

* The boresight is the vector pointing in the direction of the FOV center.
**The bounds are the set of vectors pointing to the corners of the instrument FOV.

- **cspice_reclat:** It converts rectangular coordinates (Cartesian) to latitudinal coordinates. The outputs are the distance of the position from the origin (*radius*), the angle of the position from the XZ plane (*lon*) and the angle from the XY one (*lat*).

```
rectan=[-5.256e+04;-4.836e+04;2.9115e+03];
[radius, lon, lat] = cspice_reclat(rectan);
%Command Window results
radius=7.1483e+04
lon=-3.978 % In radians
lat=.0407 % In radians
```

- **cspice_sincpt:** It computes the surface intercept of a ray, defined by an observer and a direction vector, on a target body at a specified epoch. It need multiple inputs:
  - method: String to provide the parameters that define the computation method to be used.
  - *target*: The name of the target body.
  - *et*: The epoch of the observer.
  - *fixref*: The name of a body-fixed reference frame centred on the target body.
  - *abcorr*: The aberration correction to apply when computing the observer-target state and the orientation of the target body.
  - *obsrvr*: The name of the observing body.
  - *dref*: The name of the reference frame relative to which the ray's direction vector is expressed.
  - *dvec*: The pointing vector emanating from the observer. The intercept with the target body's surface of the ray defined by the observer and *dvec* is sought.

  The function provides the following outputs:
  - *spoint*: 3-vector defining the surface intercept point on the target body of the ray.
  - *trgepc*: Epoch at which the ray defined by *obsrvr* and *dvec* intercepts the target surface at *spoint*.
  - *srfvec*: 3-vector defining the vector from the observer's position at *et* to *spoint*.
  - *found*: 0 if the ray does not intersect the target and 1 if it does.

```
method='ELLIPSOID';
target='JUPITER';
et=2.9478e+07;
fixref='IAU_JUPITER';
abcorr='LT+S';
obsrvr='CASSINI';
dref='CASSINI_ISS_NAC';
dvec=bounds; % Normally dvec has the same value as the variable bounds, obtained in the function
    cspice_getfov. However, this can vary, as in the Navigation code (dealing with pixels)
[spoint, trgepc, srfvec, found] = cspice_sincpt(method, target, et, fixref, abcorr, obsrvr, dref, dvec)
    ;
%Command Window results
```

```
spoint=[5.2150e-310;2.1962e-314;6.0842e-310]
trgepc=2.9477e+07;
srfvec=[1.1126e-306;1.6912e-306;1.4242e-306]
found=0
```

*IMPORTANT: This example is used for an specific code *Navigation* and the results provided here are obtained for a different value of *dvec*. Therefore, the outputs are only orientative.

- **cspice_ilumin:** It computes the illumination angles (phase, solar incidence and emission) at a specified surface point of a target body. The inputs are the same ones as the previous function except for the last two, which are substituted by *spoint*, which defines the surface location on the target body and is obtained from *cspice_sincpt*. Two of the five outputs are *trgepc* and *srfvec* (explained in the previous function). The other three are:
  - *phase*: It is the angle between the *spoint-obsrvr* vector and the *spoint*-sun vector.
  - *solar*: This is the angle between the surface normal vector at *spoint* and the *spoint*-sun vector.
  - *emissn*: This is the angle between the surface normal vector at *spoint* and the *spoint-obsrvr* vecotr.

```
method='ELLIPSOID';
target='JUPITER';
et=2.9478e+07;
fixref='IAU_JUPITER';
abcorr='LT+S';
obsrvr='CASSINI';
spoint=[-5.2562e+04;-4.8360e+04;2.9115e+03];
 [trgepc, srfvec, phase, solar, emissn] = cspice_ilumin(method, target, et, fixref, abcorr, obsrvr,
     spoint);
%Command Window results
trgepc=2.9477e+07;
srfvec=[-1.5876e+07;1.7390e+07;-1.4674e+06]
phase=0.1203 % In radians
solar=1.6842 % In radians
emissn=1.5641 % In radians
```

*IMPORTANT: This example is used for an specific code (*Navigation*). It also should be noted that the values in this examples does not coincide with the ones provided in the previous one, since this function is used when *found=1* in *cspice_sincpt*.

4 Errors

A list of common errors is provided below:

- If a kernel is being downloaded and, for any reason, the process is stopped or cancelled, it is necessary to find the half-downloaded kernel in the correct folder and eliminate it. On the contrary, the file will only have some data and not the whole, since the program is not able to detect that the download has been interrupted. Once the kernel is removed, it is necessary to write in the Command Window: *endSPICE* and run again the code.

- Sometimes after running the code, an error appears:

  Error using mice SPICE(SPKINSUFFDATA): Insufficient ephemeris data has been loaded to compute the position of XXX relative to YYY at the ephemeris epoch YEAR MONTH DAY TIME.

  or

  Error using mice SPICE(DAFBEGGTEND): Beginning address (XXXXXXX) greater than ending address (YYYYYYY).

  The previous messages have different meanings and can appear because of multiple reasons, but a very common reason is that the program does not have the necessary kernels to carry out the orders of the code. In these situations it is needed to check the kernels used at the beginning of the program and verify that they are the correct ones. For example, the file loaded do not contain data for both target and observer bodies or do not cover the time at which is requested a state vector [2].

- It is necessary to be careful with the order of the kernels when loading them, specially if the selection has been done manually. If one copies directly the content of the .txt file downloaded from [4] (explained in the second bullet of the next section) and loads the kernels, there should not be any problem. Nevertheless, if their order is modified, the results obtained when running a code can vary.

```
1 Spacecraft='NEW HORIZONS'
2 Encounter='PLUTO';
3 Initial_Date='2015-01-01T00:00:00';
4 Final_Date='2016-01-01T00:00:00';
```

  Among all the kernels needed to do computations between the indicated dates, there are two that can cause problems:

  `https://naif.jpl.nasa.gov/pub/naif/pds/data/nh-j_p_ss-spice-6-v1.0/nhsp_1000/data/spk/nh_plu017.bsp`

  `https://naif.jpl.nasa.gov/pub/naif/pds/data/nh-j_p_ss-spice-6-v1.0/nhsp_1000/data/spk/nh_recon_pluto_od122_v01.bsp`

  As it has been said, if one loads them in this order, which is the same one as in the .txt file, the results should be correct. Nevertheless, if *nh_plu017.bps* is placed after *nh_recon_pluto_od122_v01.bps*, the results obtained are different from the previous. To avoid this situation, it has been chosen to

load only one of them, the most important, which is *nh_recon_pluto_od122_v01.bps*, since without it an error message appears.

5 Webs of interest

Apart from the webpages and online material provided in the bibliography and referenced throughout the document, there are also other addresses related to SPICE that are useful:

- Initial page with all the missions, generic kernels and other folders of interest: `https://naif.jpl.nasa.gov/pub/naif/`.
- In order to know which specific kernels need to be downloaded when considering a particular period of time of a mission, the next link is provided: `https://naif.jpl.nasa.gov/naif/data_archived.html` [4]. In the main page, click in subset and then introduce the start and stop date (e.g. 2009 NOV 08 00:00:00.00 UTC and 2009 DEC 09 00:00:00.00 UTC). A folder with four files is downloaded and the necessary kernels are indicated in the .txt file. Then, it is possible to copy them to a Matlab script or read them from the same file.
- NAIF Integer ID codes: `https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/FORTRAN/req/naif_ids.html`.
- Classification and description of the most used MICE functions in C (the majority of functions coincide in MATLAB): `https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/C/info/mostused.html`
- Aberration correction types and definitions: `https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/C/req/abcorr.html`.

Bibliography

[1] *ISS Instrument Kernel (Cassini)*. URL: https://naif.jpl.nasa.gov/pub/naif/CASSINI/kernels/ik/cas_iss_v10.ti.

[2] Navigation and Ancillary Information Facility. *Ephemeris Subsystem SPK*. Errors starting on page 57. Jan. 2020. URL: https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/Tutorials/pdf/individual_docs/18_spk.pdf.

[3] Navigation and Ancillary Information Facility. *Index of MICE functions*. URL: https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/MATLAB/mice/index.html.

[4] *PDS SPICE Archives*. Mar. 2021. URL: https://naif.jpl.nasa.gov/naif/data_archived.htm.

A Other MICE functions

- **cspice_axisar:** It returns a 3x3 double rotation matrix that rotates vectors by a specified angle about a specified axis.
- **cspice_bodc2s:** It translates a body ID code to either the corresponding name or if no name to ID code mapping exists, the string representation of the body ID value. It is a more general version of *cspice_bodn2c*.
- **cspice_bodfnd:** It determines whether values exist for some item for any body in the kernel pool.
- **cspice_bodvcd:** It has the same purpose as *cspice_bodvrd*, but in this case the input has to be the ID code of the body and not its name.
- **cspice_ckobj:** It returns the set of ID codes of all objects in a specified CK file.
- **cspice_cnmfrm:** It retrieves the ID code and name of the preferred frame associated with a given body name.
- **cspice_convrt:** It performs a conversion from a measurement in one unit set to the corresponding measure in another unit set.
- **cspice_cyllat:** It converts cylindrical coordinates to latitudinal coordinates.
- **cspice_cylrec:** It converts cylindrical coordinates to rectangular (Cartesian) coordinates
- **cspice_cylsph:** It converts cylindrical coordinates to spherical coordinates.
- **cspice_dskobj:** It returns the set of body ID codes of all objects for which topographic data are provided in specified DSK files.
- **cspice_dskxsi:** It computes a ray-surface intercept using data provided by multiple loaded DSK segments. Return information about the source of the data defining the surface on which the intercept was found: DSK handle, DLA and DSK descriptors, and DSK data type-dependent parameters.
- **cspice_edlimb:** It calculates the limb of a triaxial ellipsoid as viewed from a specified location.
- **cspice_edterm:** It computes a set of points on the umbral or penumbral terminator of a specified target body, where the target shape is modeled as an ellipsoid.
- **cspice_fovray:** It determines if a specified ray is within the field-of-view (FOV) of a specified instrument at a given time.
- **cspice_frmnam:** It retrieves the name of a reference frame associated with a SPICE frame ID code.
- **cspice_gcpool:** It returns the value of a string kernel variable (scalar or array) from the kernel pool. It has the same estructure as *cspice_gdpool* and *cspice_gipool*.
- **cspice_gnpool:** It returns the names of kernel variables matching a specified template.

- **cspice_georec:** It converts geodetic coordinates to rectangular coordinates.
- **cspice_gfdist:** It determines the time intervals over which a specified constraint on observer-target distance is met.
- **cspice_gfoclt:** It determines time intervals when an observer sees one target body occulted by, or in transit across, another.
- **cspice_gfrfov:** It determine time intervals when a specified ray intersects the space bounded by the field-of-view (FOV) of a specified instrument.
- **cspice_gftfov:** It determines time intervals when a specified ephemeris object intersects the space bounded by the field-of-view (FOV) of a specified instrument.
- **cspice_illum:** It calculates the illumination angles at a specified surface point of a target body.
- **cspice_illumf:** It computes the illumination angles-phase, incidence, and emission-at a specified point on a target body. Return logical flags indicating whether the surface point is visible from the observer's position and whether the surface point is illuminated.
- **cspice_illumg:** It is similar to *cspice_illumf*, but it does not logical flags.
- **cspice_kclear:** It clears the KEEPER system: unload all kernels, clears the kernel pool, and re-initialize the system.
- **cspice_kdata:** It returns data for the nth kernel among a list of specified kernel types.
- **cspice_kinfo:** It returns information about a loaded kernel specified by name.
- **cspice_ktotal:** It returns the current number of kernels loaded.
- **cspice_latcycl:** It converts from latitudinal coordinates to cylindrical coordinates.
- **cspice_latrec:** It converts latitudinal coordinates to rectangular (Cartesian) coordinates.
- **cspice_latsph:** It converts latitudinal coordinates to spherical coordinates.
- **cspice_latsrf:** It maps an array of planetocentric longitude/latitude coordinate pairs to surface points on a specified target body.
- **cspice_limbpt:** It finds limb points on a target body. The limb is the set of points of tangency on the target of rays emanating from the observer. The caller specifies half-planes bounded by the observer-target center vector in which to search for limb points.
- **cspice_namfrm:** It retrieves the SPICE frame ID code associated with a frame name.
- **cspice_nearpt:** It calculates the point on the surface of an ellipsoid nearest to a specified off-ellipsoid position.
- **cspice_npedln:** It calculates the nearest point on a triaxial ellipsoid to a specified line, and the distance from the ellipsoid point to the line.

- **cspice_npelpt** It calculates the location on an ellipse closest to a specified point, both in three-dimensional space, and the distance between the ellipse and the point.
- **cspice_nplnpt:** It calculates the location on a defined line nearest to a specified point, then determines the distance between the two points.
- **cspice_occult:** It determines the occultation condition (not occulted, partially, etc.) of one target relative to another target as seen by an observer at a given time.
- **cspice_oscelt:** It calculates the set of osculating conic orbital elements corresponding to the state 6-vector (position, velocity) of a body at an epoch.
- **cspice_pckcov:** It returns the coverage window for a specified reference frame in a specified binary PCK file.
- **cspice_pckfrm:** It returns the set of reference frame class ID codes of all frames in a specified binary PCK file.
- **cspice_pgrrec:** It converts planetographic coordinates to rectangular coordinates.
- **cspice_radrec:** It converts the right ascension, declination coordinates of a location to rectangular (Cartesian) coordinates.
- **cspice_reccyl:** It converts rectangular (Cartesian) coordinates to cylindrical coordinates.
- **cspice_recgeo:** It converts rectangular coordinates to geodetic coordinates.
- **cspice_recpgr:** It converts rectangular coordinates to planetographic coordinates.
- **cspice_recrad:** It converts rectangular (Cartesian) coordinates to right ascension, declination coordinates.
- **cspice_recsph:** It converts rectangular (Cartesian) coordinates to spherical coordinates.
- **cspice_scdecd:** It converts a double precision encoding of spacecraft clock time into a string representation.
- **cspice_sphcyl:** It converts spherical coordinates to cylindrical coordinates.
- **cspice_sphlat:** It converts spherical coordinates to latitudinal coordinates.
- **cspice_sphrec:** It converts spherical coordinates to rectangular (Cartesian) coordinates.
- **cspice_srfrec:** It converts planetocentric latitude and longitude of a surface point on a specified body to rectangular coordinates.
- **cspice_surfnm:** It computes the double precision, outward-pointing normal unit 3-vector at a point defined on the surface of an ellipsoid.
- **cspice_termpt:** It finds terminator points on a target body. The terminator is the set of points of tangency on the target body of planes tangent to both this body and to a light source. The caller specifies half-planes, bounded by the illumination source center-target center vector, in which to search for terminator points.

- **cspice_tpictr:** It takes a time format sample then creates a time format picture suitable for use by *cspice_timout*.
- **cspice_vdist:** It returns the distance between two three-dimensional vectors.
- **cspice_vhat:** It returns the unit vector along a double precision 3-dimensional vector.
- **cspice_vnorm:** It returns the magnitude of a double precision, 3-dimensional array or set of such arrays.
- **cspice_vperp:** It calculates the component of a vector perpendicular to a second vector.
- **cspice_unorm:** It normalizes a double precision 3-vector and returns its magnitude.
- **cspice_unload:** It unloads a SPICE kernel file (of any type) from MATLAB.
- **cspice_xfmsta:** It transforms a state between coordinate systems.