Handbook on "SPICE for MATLAB"

Mauro De Francesco[*], Erind Veruari[†], Gianmario Merisio[‡]
Politecnico di Milano, 20156 Milan, Italy
February 2020

This document provides the following content: 1) Introduction to SPICE, 2) Initialization of SPICE for MATLAB, 2) Introduction to the concept of Kernels, 4) Introduction to the most common APIs, and 5) Useful Links. Instructions are provided for Mac, Linux Ubuntu distribution, and Windows OS.

[*] MSc Student, Dept. of Aerospace Science and Technology.
[†] MSc, Dept. of Aerospace Science and Technology.
[‡] PhD Student, Dept. of Aerospace Science and Technology; gianmario.merisio@polimi.it.

1 Introduction to SPICE

SPICE is an ancillary information system that provides the capability to include space geometry and event data into mission design, science observation planning, and science data analysis software. The use of SPICE extends from mission concept development through the post-mission data analysis phase, including help with correlation of individual instrument data sets with those from other instruments on the same or on a different spacecraft. For any additional information about SPICE visit the NAIF website.

The principle components of the SPICE system are SPICE Toolkit software and SPICE data files, often called kernels. It is possible to download the SPICE Toolkit in different programming languages. The one used for this handbook is MATLAB and the relative toolkit is called **mice**.

In Sec. 2 the instructions to initialize mice are given. Then, in Sec. 3 is reported a general description of kernels. Finally, in Sec. 4 are reported the most used functions offered in the SPICE Toolkit.

2 Initialization of SPICE for MATLAB

2.1 *Windows OS: Download and Initialization*

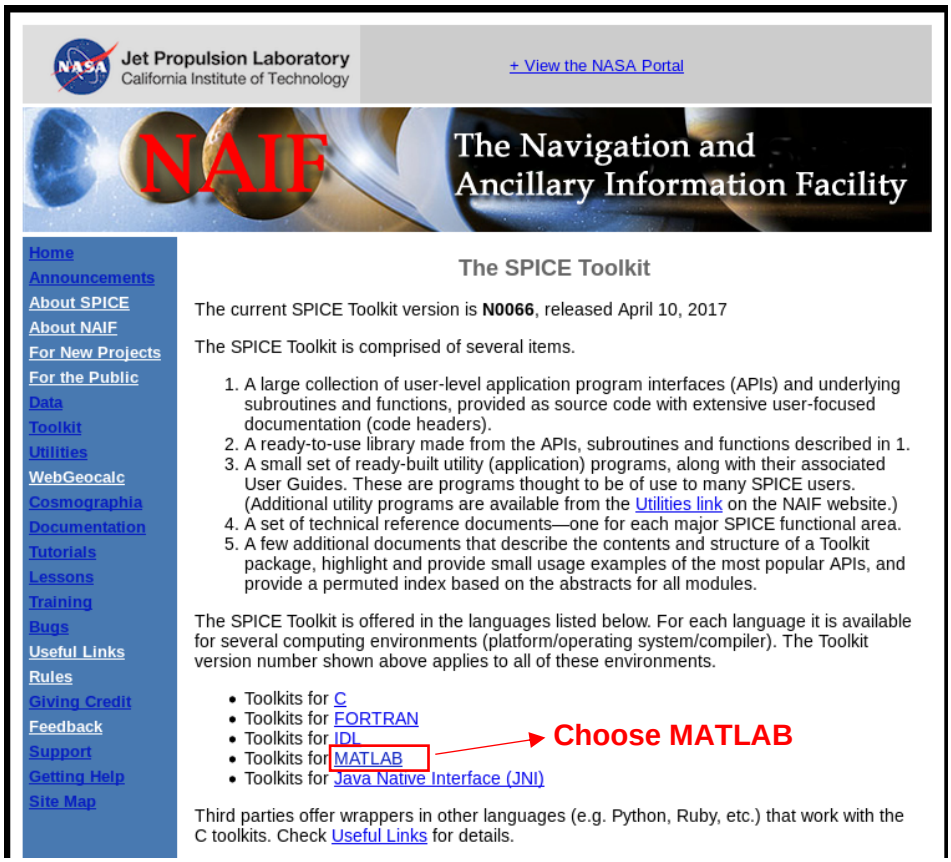
To initialize mice the following steps are required:

1) Download mice;
2) Initialize mice;

2.1.1 *Download mice*

To download mice the steps are the following:

1) Go to the toolkit download page.
2) Choose the language. Different languages are available: C, Fortran, IDL, MATLAB, and JNI. Here the MATLAB language is used.



3) Choose the platform. Different platforms are compatible with the toolkit: Mac, Linux, Windows, etc. Select the toolkit compatible with your platform.

4) Select and download the file *mice.zip*.

5) Choose the path where to download the *mice.zip* file. Note that the path can be whatever you want, even not related with MATLAB environment.
6) Unzip the file. It is already in a folder called *mice*.

2.1.2 Initialize mice

The use of mice requires both the *lib* and mice *src* directories exist in the MATLAB search path. The inclusion of the miice directories to the MATLAB search path can be done programmatically running in the MATLAB command window the following:

```
>> addpath('c:\path_to_mice\mice\src\mice\')
>> addpath('c:\path_to_mice\mice\lib\')
```

Where `\path to mice\` must be substituted by the path where your mice has been collocated. Then, you can use the MATLAB function `'savepath'` to save the paths permanently.

Differently, a user can also add the mice directories to the MATLAB search path by setting the MATLABPATH environment variable or creating a script that must be called every time MATLAB is opened. For additional information about these procedures refer to the mice and MATLAB documentation.

2.2 Mac OS and Linux OS: Download and Initialization

To initialize mice the following steps are required:

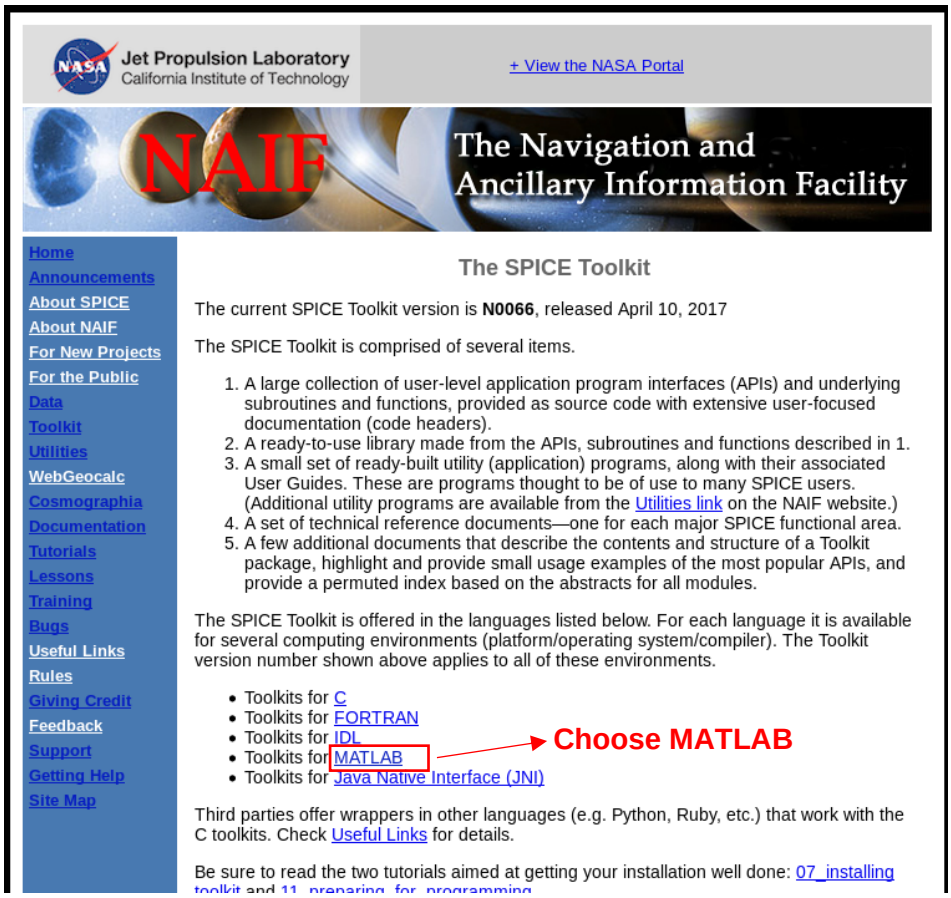
1) Download mice;
2) Initialize mice.

2.2.1 Download mice

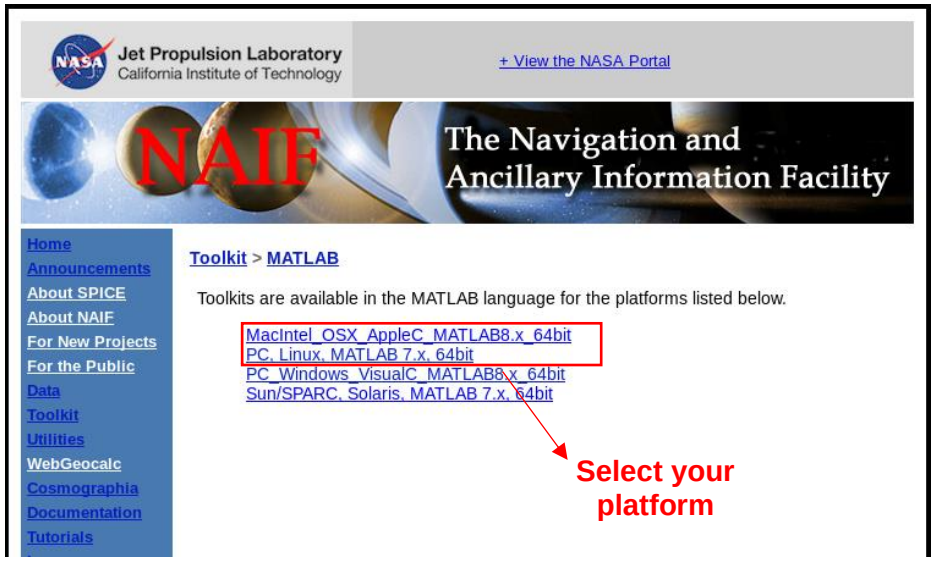
To download mice the steps are the following:

1) Go to the toolkit download page.

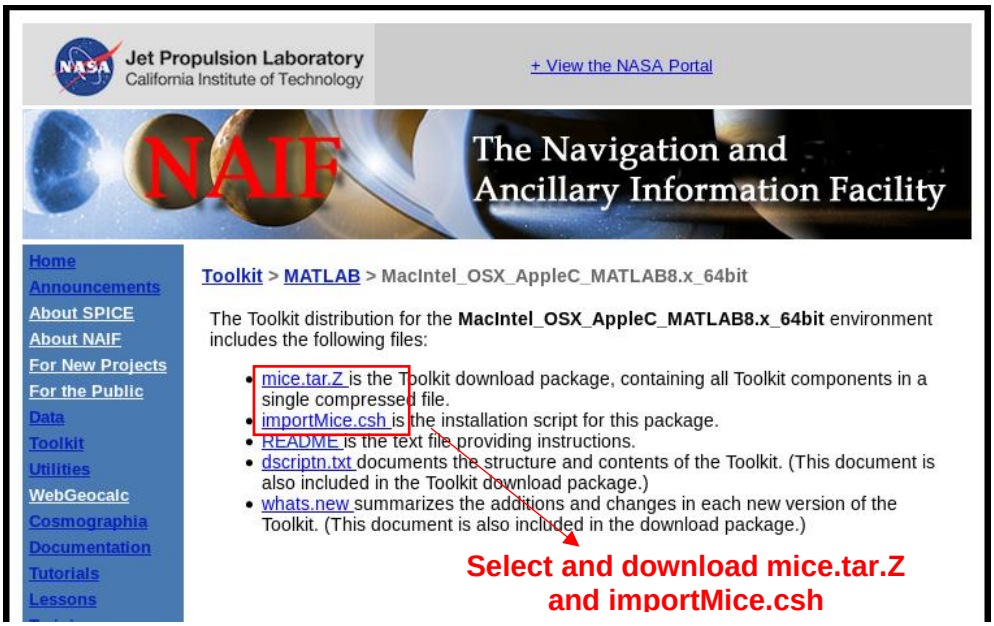
2) Choose the language. Different languages are available: C, Fortran, IDL, MATLAB, and JNI. Here the MATLAB language is used.



3) Choose the platform. Different platforms are compatible with the toolkit: Mac, Linux, Windows, etc. Select the toolkit compatible with your platform.



4) Select and download the file *mice.tar.Z* and the installation script *importMice.csh* needed to import the mice package.



5) Choose the path where to download the files. Note that the path can be whatever you want, even not related with MATLAB environment.
6) Now go to the directory where you downloaded the files and type the following command:

```
/bin/csh -f importMice.csh
```

The script *importMice.csh* will uncompress and untar the toolkit and, on platforms where NAIF anticipates that it is necessary, compile and link all source code products.

2.2.2 *Initialize mice*

The use of mice requires both the *lib* and mice *src* directories exist in the MATLAB search path. The inclusion of the miice directories to the MATLAB search path can be done programmatically running in the MATLAB command window the following:

```
>> addpath('/path_to_mice/mice/src/mice/')
>> addpath('/path_to_mice/mice/lib/')
```

Where `/path_to_mice/` must be substituted by the path where your mice has been collocated. Then, you can use the MATLAB function `'savepath'` to save the paths permanently.

Differently, a user can also add the mice directories to the MATLAB search path by setting the MATLABPATH environment variable or creating a script that must be called every time MATLAB is opened. For additional information about these procedures refer to the mice and MATLAB documentation.

2.3 *Test mice*

To ensure a proper setup, execute the MATLAB command:

```
>> which mice
```

MATLAB should return the path to the *mice.mex** file if the file exists within a directory searched by MATLAB.

```
/path_to_mice/mice/lib/mice.mex*
```

If MATLAB outputs:

```
'mice' not found.
```

Then the MATLAB search path does not include the directory containing the *mice.mex** file.

The Matlab command:

```
>> cspice_tkvrsn('toolkit')
```

causes MATLAB to display the string identifier for the CSPICE library version (N00XX) against which mice is linked.

2.4 *Download Basic Kernels*

To have SPICE correctly work you need specific kernels or files. The concept of kernel will be introduced later in the handbook. NAIF refers as a combination of standard kernels to the triad composed by:

- **Leapseconds Kernel**
  - Used for time conversion
  - Called LSK
  - The file extension is **.tls*
  - Download page (choose *naif0012.tls*)
- **Spacecrafts and Planet Kernel**:
  - Used for the ephemerides of bodies
  - Called SPK
  - The file extension is **.bsp*
  - Download page (choose *de432s.bsp*)
- **Planetary Constants Kernel**:
  - Used for frame identification and containing planetary constants
  - Called PCK
  - The file extension is **.tpc*
  - Download page (choose *pck00010.tpc*)

3 Kernels

A *kernel* is a file containing "low level" ancillary data that may be used, along with other data and SPICE Toolkit software, to determine higher level observation geometry parameters of use to scientists and engineers in planning and carrying out space missions, and analyzing data returned from missions.

3.1 Types of Kernel

There are various types of kernel:

- **LSK** - *leapseconds kernels*: they are used in conversions between ephemeris time (ET/TDB) and Coordinated Universal Time (UTC).
- **SPK** - *spacecrafts and planets kernels*: they contain vehicles, planets, satellites, comets, asteroids ephemeris (position and velocities).
- **PCK** - *planetary constants kernels*: they contain planetary constants for natural bodies (orientation, size and shape).
- **IK** - *instrument kernels*: they contain instrument-specific geometry information (field-of-view, size, shape, orientation, etc.).
- **CK** - *camera-matrix kernels*: they hold orientation (attitude) data of a spacecraft or moving structure.
- **EK** - *event kernels*: they are a collection of information concerning planned or unplanned mission activities or occurrences that can assist in extracting the full value of the science data returned from those missions.
- **FK** - *frame kernels*: they establish relationships between reference frames used in geometry computations.
- **SCLK** - *spacecraft clock kernels*: they are used in conversions between spacecraft clock time (SCLK) and ephemeris time (ET/TDB).
- **DSK** - *digital shape kernels*: they contain detailed shape models for extended objects.

3.2 How to Load Kernels

Kernels must be loaded within the so-called *kernel pool* in order to be successfully used by SPICE. There is a specific function that can be used to load kernels. The function is `'cspice_furnsh'`. In order to load them, kernels are given as input to such function. For instance, to load the leapsecond kernel LSK the call is the following:

```
cspice_furnsh({'/path_to_kernels/naif0012.tls'})
```

In order to load all the kernels needed in one-shot, it is possible to use a special type of kernel named *Meta-Kernel* (MK). A MK is a text file that lists names and locations of a collection of SPICE kernels that need to be used together in a SPICE-based application. A MK example follows:

```
\begintext
The command "\begintext" is interpreted by SPICE as a comment in the meta-
kernel file.
The function cspice_furnsh skips this part of the text.
The command "\begindata" tells SPICE that the text after contains the data to
load.
Use the format below to insert all the kernels to load:
KERNELS_TO_LOAD = ('..path1/kernel1.extention1',
                   '..path2/kernel2.extention2',
                   ...)

\begindata
KERNELS_TO_LOAD = ('kernels/naif0012.tls',
                   'kernels/naif0012.lbl',
                   'kernels/pck00010.tpc',
```

```
                    'kernels/gm_de431.tpc',
                    'kernels/oblateness.tpc',
                    'kernels/de432s.bsp',
                    'kernels/earth_000101_161224_161002.bpc',
                    'kernels/Halo_Cj3p09/Halo_Cj3p09_LONG.bsp',
                    'kernels/Halo_Cj3p09/Halo_Cj3p09.bsp')

\begintext
Multiple "\begintext" and "\begindata" can be used in the same text file.
```

After the MK is written, you can give it as input to the `'cspice_furnsh'` as you would do with any other kernel.

3.3 Download Kernels

Download the kernels released by NAIF from the official page SPICE Data.

Note that it is possible to make your own kernels if you need data which are not present in the generic kernels, but that is not covered in this handbook. Refer to the official SPICE documentation if you need to write your own kernels.

4 Most Common APIs

In this section are reported and described the most common SPICE functions. Each function name is a link to the SPICE webpage where a more in-depth description is present. At mice APIs it is possible to find the list and the helps of all the functions available within the mice toolkit.

4.1 *General*

- cspice_furnsh: loads SPICE kernel files into MATLAB.
- cspice_ktotal: returns the current number of kernels of a specific type loaded via cspice_furnsh.
- cspice_kdata: returns data for the nth kernel among a list of specified kernel types.
- cspice_gcpool: returns the value of a string kernel variable (scalar or array) from the kernel pool.
- cspice_gdpool: returns the value of a double precision kernel variable (scalar or array) from the kernel pool.
- cspice_kclear: clears the KEEPER system, which unloads all kernels, clears the kernel pool, and reinitializes the system.

4.2 *Time Conversion*

- cspice_timout: 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.
- cspice_str2et: 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.
- cspice_et2utc: converts an input time from ephemeris seconds past J2000 to Calendar, Day-of-Year, or Julian Date format, UTC.
- cspice_etcal: converts from an ephemeris epoch measured in seconds past the epoch of J2000 to a calendar string format using a formal calendar free of leapseconds.

4.3 *Object Information*

- cspice_spkpos: returns the position of a target body relative to an observing body, optionally corrected for light time (planetary aberration) and stellar aberration.
- cspice_spkezr: 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.
- cspice_ckcov: returns the coverage windows for a specified object in a specified CK file.
- cspice_ckgpav: returns pointing (attitude) and angular velocity for a specified spacecraft clock time.
- cspice_bodvrd: fetches from the kernel pool the double precision values of an item associated with a body.

4.4 *Frames*

- cspice_sxform: returns the state transformation matrix from one frame to another at a specified epoch.
- cspice_pxform: returns the matrix that transforms position vectors from one specified frame to another at a specified epoch.
- cspice_pxfrm2: returns the 3x3 matrix that transforms position vectors from one specified frame at a specified epoch to another specified frame at another specified epoch.

4.5 *Illumination*

- cspice_illumf: computes the illumination angles (phase, solar incidence, and emission) at a specified surface point on a target body. It returns logical flags indicating whether the surface point is visible from the observer's position and whether the surface point is illuminated.
- cspice_illumg: computes the illumination angles (phase, solar incidence, and emission) at a specified surface point of a target body. The surface of the target body may be represented by a triaxial ellipsoid or by topographic data provided by DSK files. The illumination source is a specified ephemeris object.
- cspice_illumin: computes the illumination angles (phase, solar incidence, and emission) at a specified surface point of a target body.
- cspice_subslr: computes the rectangular coordinates of the sub-solar point on a target body at a specified epoch, optionally corrected for light time and stellar aberration. The surface of the target body may be represented by a triaxial ellipsoid or by topographic data provided by DSK files.
- cspice_dskxsi: computes a ray-surface intercept using data provided by multiple loaded DSK segments. It returns 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_lspcn: computes the planetocentric longitude of the sun, as seen from a specified body.
- cspice_phaseq: computes the apparent phase angle for a target, observer, illuminator set of ephemeris objects.
- cspice_limbpt: 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_termpt: 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. The terminator can be either umbral or penumbral. The umbral terminator is the boundary of the region on the target surface where no light from the source is visible. The penumbral terminator is the boundary of the region on the target surface where none of the light from the source is blocked by the target itself. The surface of the target body may be represented either by a triaxial ellipsoid or by topographic data.

5 Useful Links

- NAIF Homepage: here you may access all official SPICE products produced by NAIF.
- SPICE Tutorials: here you may access a collection of tutorials in a presentation format, covering most aspects of using SPICE.
- SPICE Data: here you may access all the SPICE kernels available from NAIF portal.
- mice Download: here you can download your mice toolkit. Inside the folder downloaded you will have a **{.../doc/html}** where you can find the official documentation.