SPICE-based Python packages
for Solar System Exploration geometry exploitation

Marc Costa
ESA SPICE Service
Markus Grass
TU Stuttgart

ESPC Conference
17th September, Berlin, Germany

What will be presented today

- SPICE in a nutshell (very brief intro to SPICE)
- WebGeocalc & Cosmographia
- A Working example
- Pros and Cons of using SPICE
- spiops

SPICE in a nutshell

SPICE is an information system that uses *ancillary data* to provide Solar System geometry information to scientists and engineers for planetary missions in order to plan and analyze scientific observations from space-born instruments. SPICE was originally developed and maintained by the Navigation and Ancillary Information Facility (NAIF) team of the Jet Propulsion Laboratory (NASA).

"Ancillary data" are those that help scientists and engineers determine:

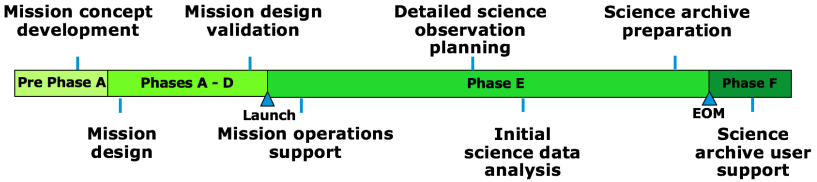
where the **spacecraft** was **located**

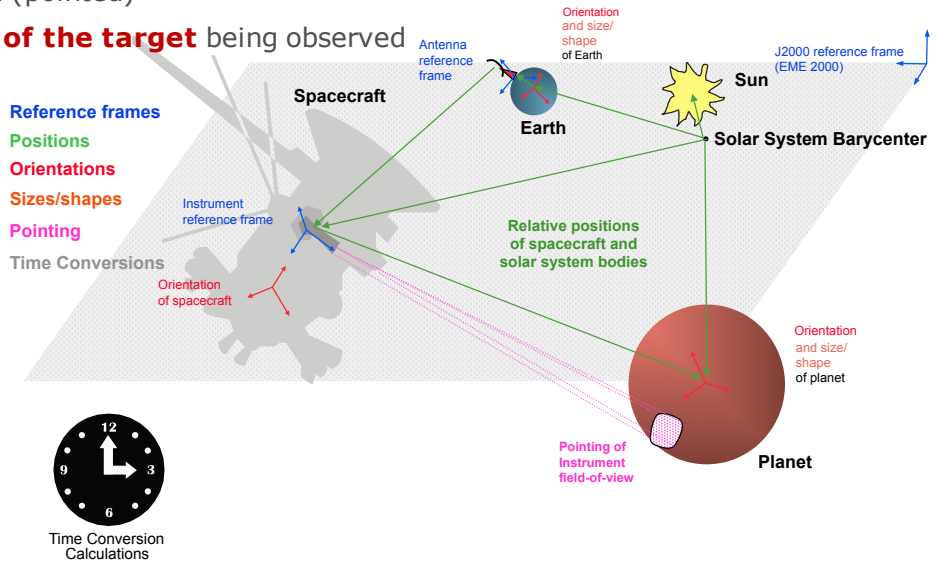
how the spacecraft and its instruments were **oriented** (pointed)

what was the **location, size, shape and orientation of the target** being observed

what **events were occurring** on the spacecraft

- SPICE provides users a large suite of SW used to read SPICE ancillary data files to compute observation geometry.
- The ancillary data (kernels) comes from: The S/C, MOC/SGS, S/C manufacturer and Instrument teams, Science Organizations.

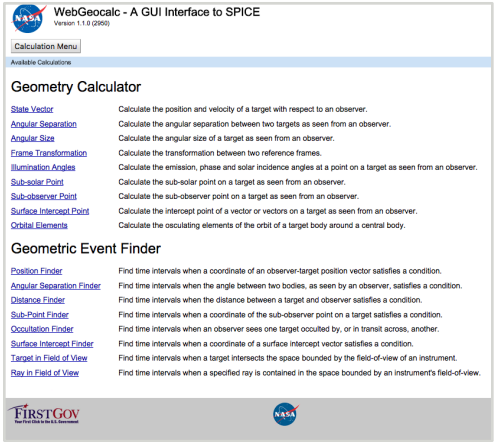


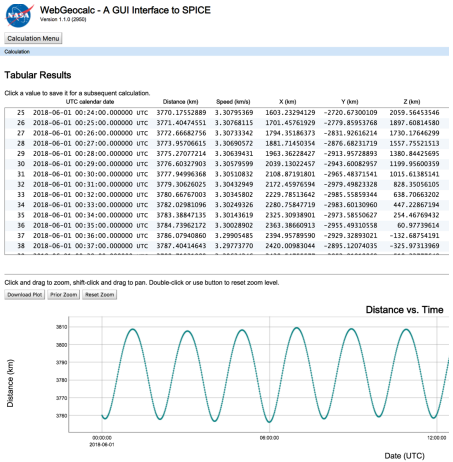


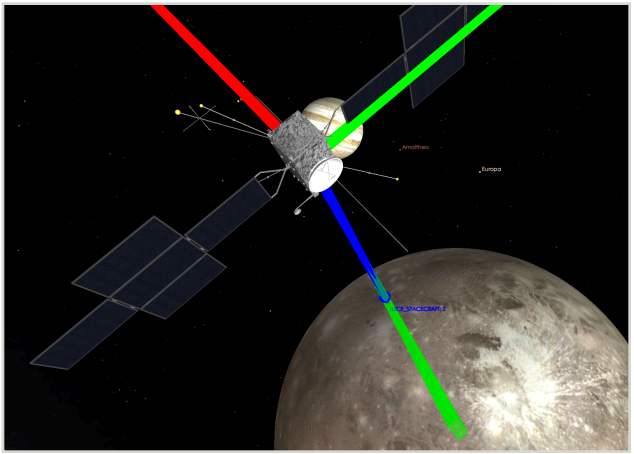
WebGeocalc & Cosmographia

WebGeocalc (WGC) is a web-based graphical user interface to SPICE. Many observation geometry computations available in SPICE through a standard web browser which provides an interface to some of the SPICE APIs.

The ESS offers an instance of WebGeocalc for the operational, studies and archived scenarios of the supported missions.







SPICE-enhanced Cosmographia is an interactive tool; 3D visualization of S/C trajectory, orientation and instrument field-of-views and footprints. It is an excellent tool for the visualization of science observations, it also provides angles, vectors, etc.

The ESS offers mission-specific configuration for Cosmographia along with training and support. Cosmographia is available at the NAIF Website.

A working example

- We want to analyze Phobos images from the HRSC instrument in MEX, more concretely images that with good resolution taken less than 1.000 km from Phobos.
- Then we could constrain our search in the PSA UI.

Input Values

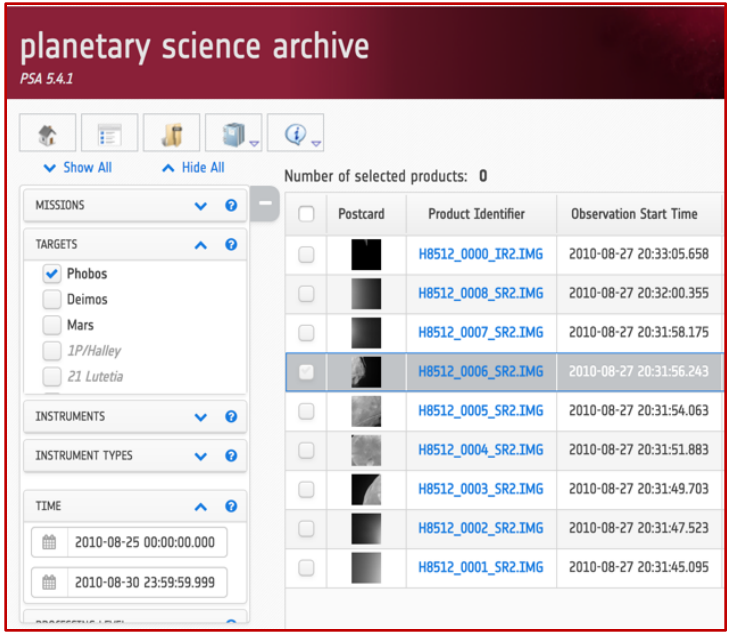
Calculation type	Distance Event Finder
Target	PHOBOS
Observer	MARS EXPRESS
Light propagation	No correction
Time system	UTC
Time format	Calendar date and time
Time range	2010-01-01 to 2010-12-01, step 6 hours
Event condition	is less than 1000
Output time unit	seconds
Complement result window	no
Result interval adjustment	No adjustment
Result interval filtering	No filtering

Tabular Results

Click a value to save it for a subsequent calculation.

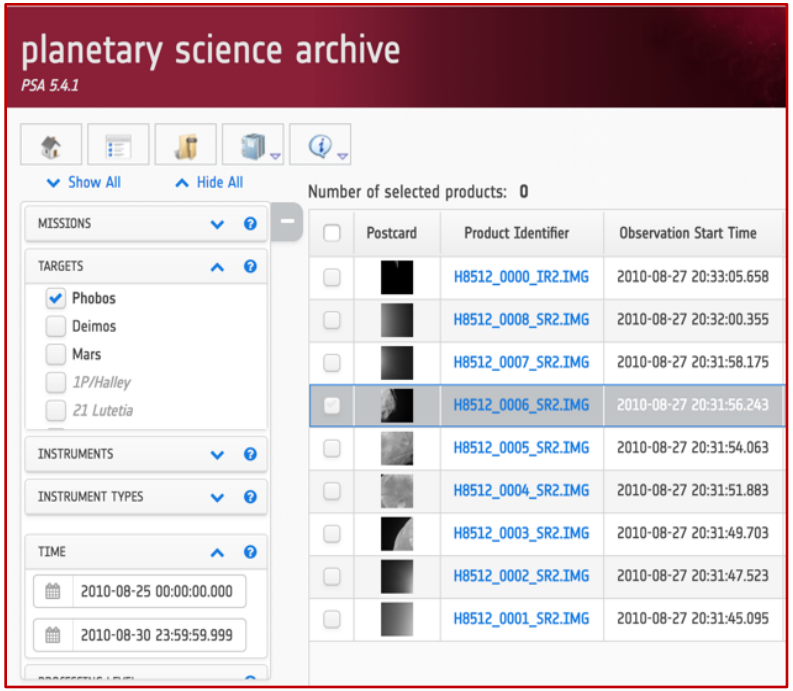
Save All Intervals

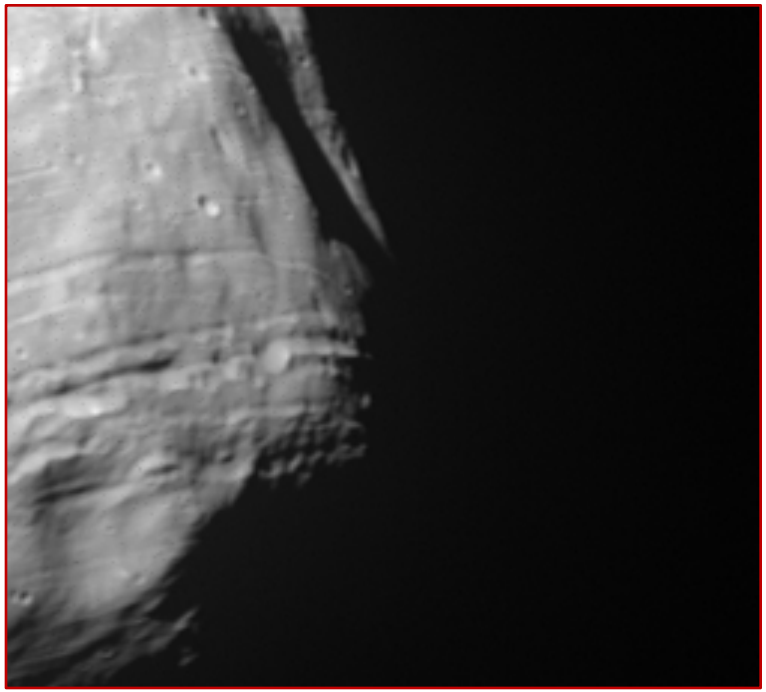
	Start Time	Stop Time	Duration (secs)
1	2010-02-28 16:18:07.102645 UTC	2010-02-28 16:29:03.406319 UTC	656.30367434
2	2010-08-24 08:22:05.025171 UTC	2010-08-24 08:32:29.283596 UTC	624.25842512
3	2010-08-27 20:28:51.926715 UTC	2010-08-27 20:34:53.806208 UTC	361.87949306



A working example

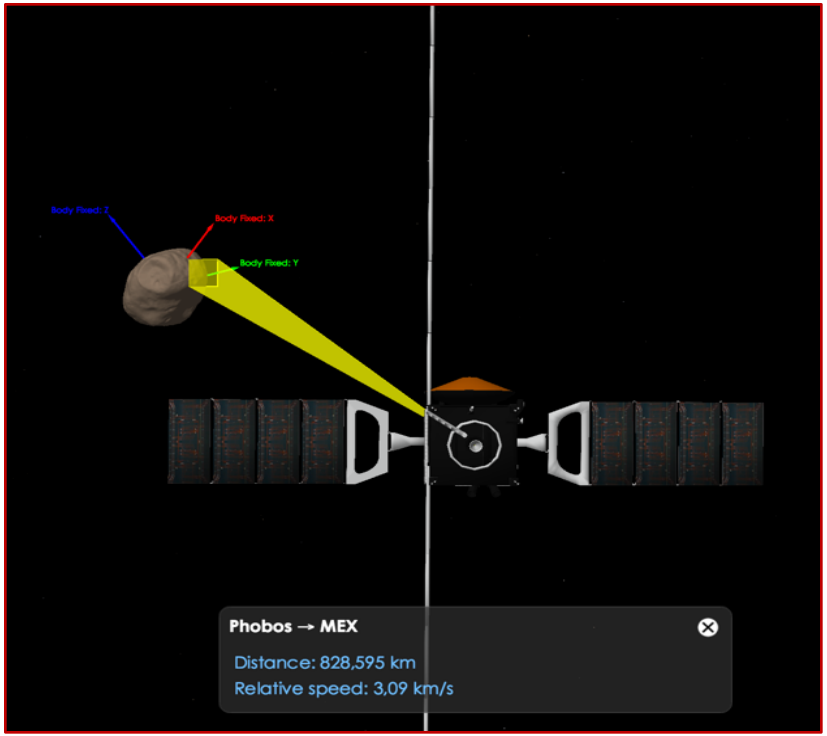
- We want to analyze Phobos images from the HRSC instrument in MEX, more concretely images that with good resolution taken less than 1.000 km from Phobos
- Then we could constrain our search in the PSA UI.

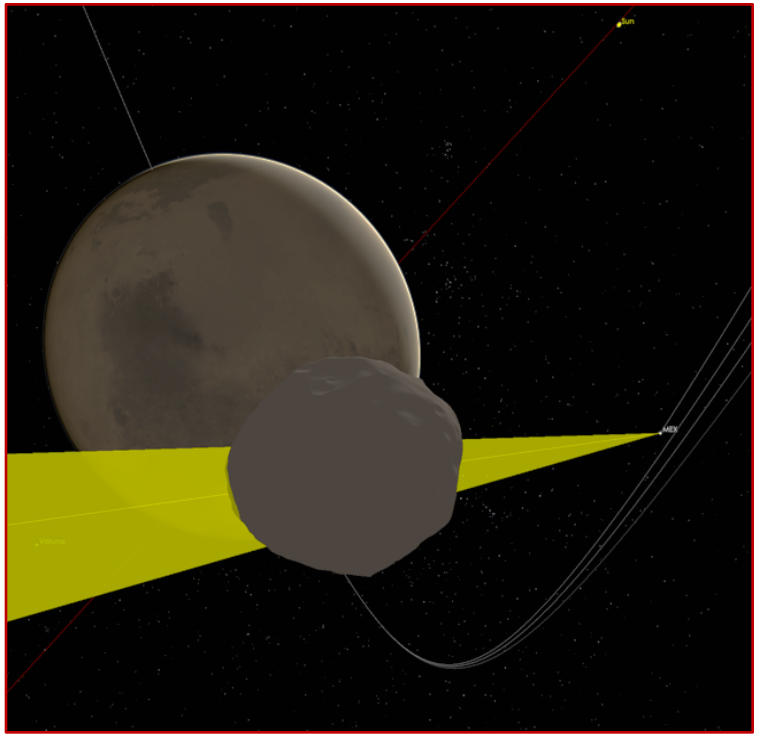




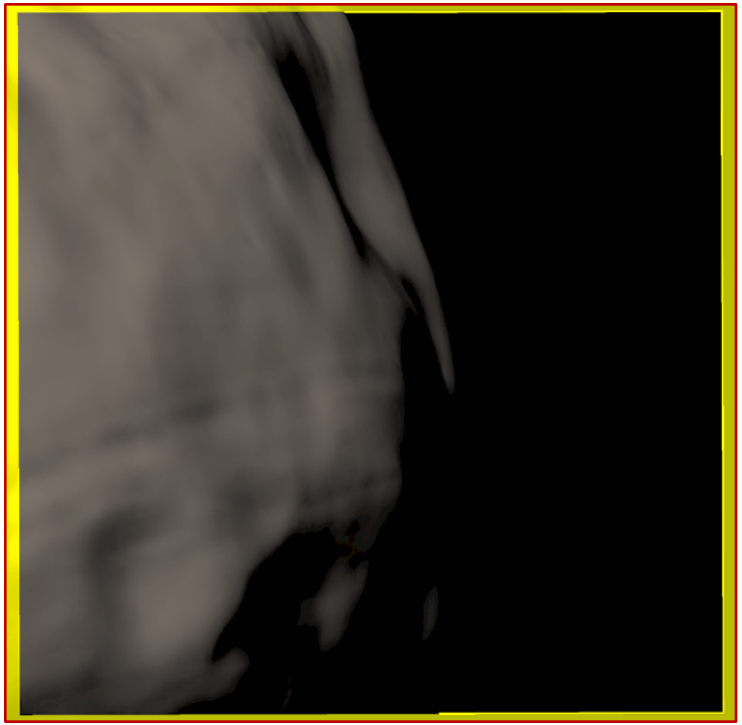
A working example

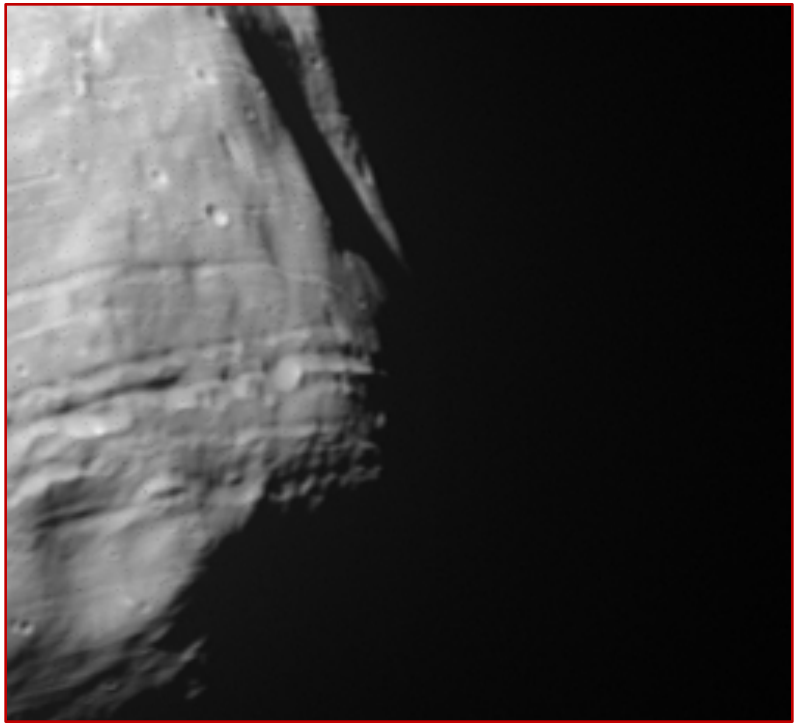
- We can use Cosmographia to asses the geometry of the observation and to double-check that the kernels are correct





A working example



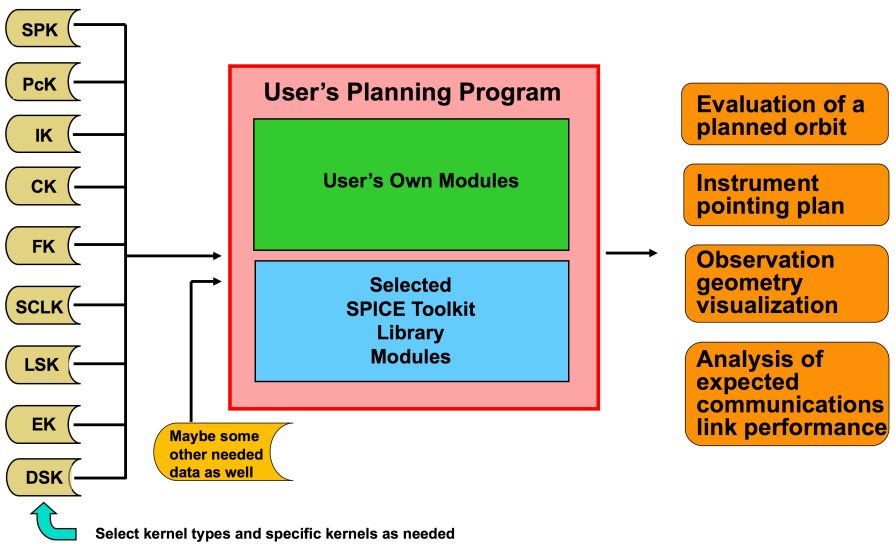


- You can see more on the working example in the SPICE seminar on Wednesday afternoon.

Using SPICE

Using SPICE as a Library requires reasonable level for the given programming language and some basic functionalities need considerable development effort e.g.:

- Loops always required for time intervals
- Nesting several APIs to obtain a certain derived geometry quantities E.g.: obtaining coverage
- Repeat multiple times the same function
- Repeat code that has been written in the past



In order to facilitate using SPICE:

- Why not gather the day-to-day SPICE-based functions and put them into a Python Package?
- Why not use the package to interface with users to generate certain geometry quantities?
- Why not share all those functions and allow users to add their own?
- Why not develop and object oriented library that allows even for generating those plots that I seem to need recursively?

spiops

spiops is **a Python package** that uses **SpiceyPy** to use SPICE Toolkit APIs to provide higher-level functions than the ones available with SPICE.

spiops is aimed to assists the users to extend the usage of SPICE.

These functions have been identified in my day-to-day work from having to implement multiple times a series of SPICE APIs to obtain a given derived functionality. **Functionalities** vary from the computation of the illumination of a given Field-of-View to obtaining the coverage of a given S/C for a particular meta-kernel, plotting Euler Angles or comparing different kernels.

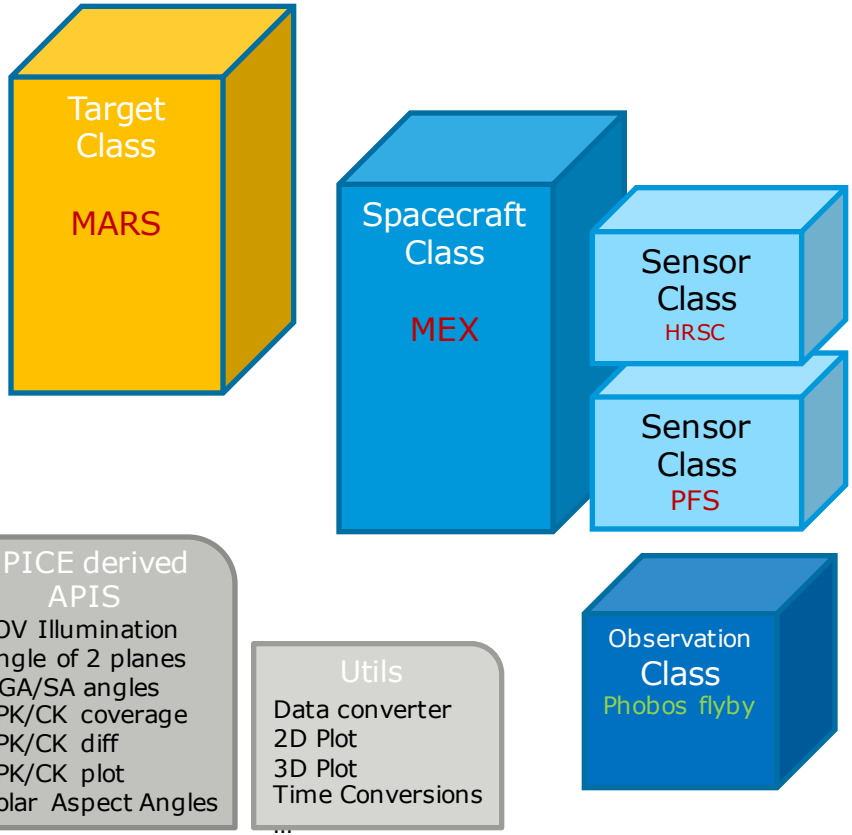
There are three different levels of functions used:

1. SPICE based derived functions
2. non-SPICE based derived functions
3. Object Oriented SPICE interface

The underlying idea of spiops is to be used as a **multi-user and multi-disciplinary pool of re-usable SPICE based functions** and to provide an easier interface to certain SPICE functionalities with objects to provide cross mission and discipline support of SPICE for ESA Planetary and Helio-phyisics missions.

Will be available as a PyPi package and accessible via BitBucket

spiops


Target Class
MARS
Spacecraft Class
MEX
Sensor Class
HRSC
Sensor Class
PFS
Observation Class
Phobos flyby
SPICE derived APIS
FOV Illumination
Angle of 2 planes
HGA/SA angles
SPK/CK coverage
SPK/CK diff
SPK/CK plot
Solar Aspect Angles
Utils
Data converter
2D Plot
3D Plot
Time Conversions
...
PROPERTIES
Trajectory, Orientation
HGA, SA, Structures
PROPERTIES
Trajectory, Orientation
HGA, SA, Structures
PROPERTIES
FOV shape, corners
Boresight
Optical parameters
PROPERTIES
Current time
Start Time
End Time
External Data Layout
METHODS
setSate
setOrientation
setGeometry
getPlot
Angles
Distances
....
getPlot3D
getEve
nts

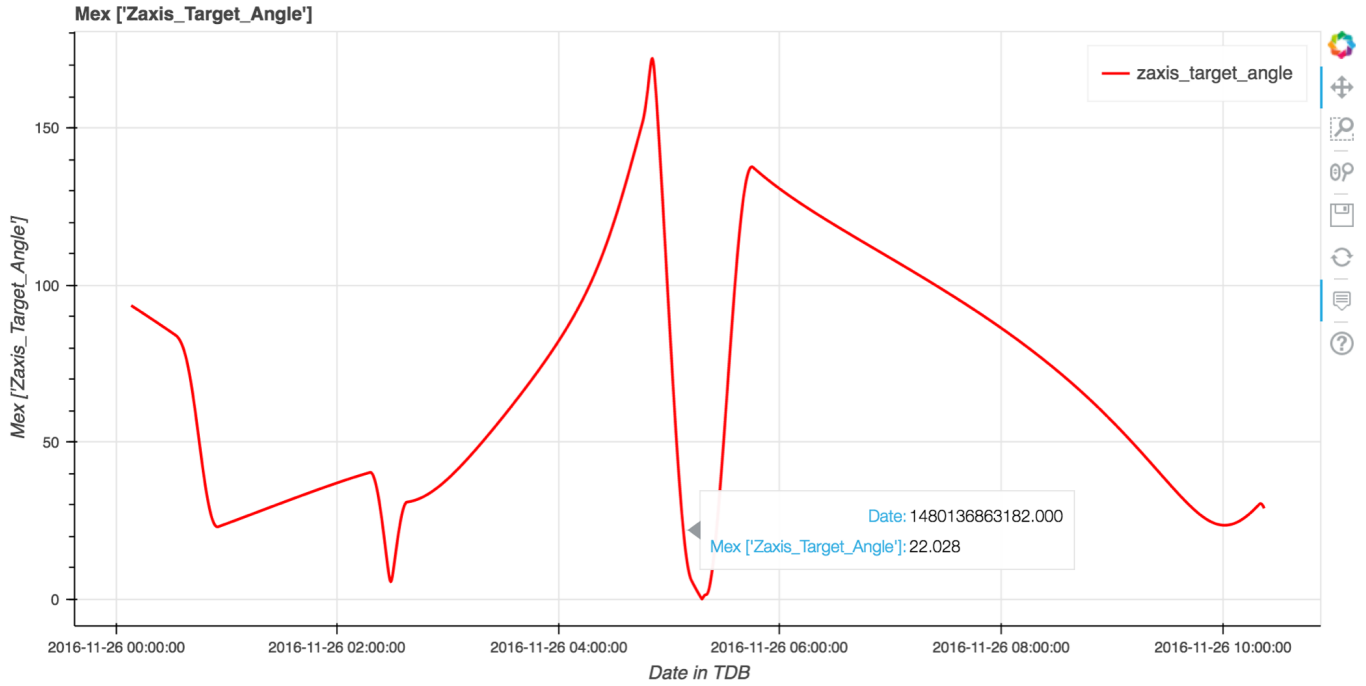
spiops

```
import spiops as spiops
from spiops.utils import utils

interval = spiops.TimeWindow('2016-11-26T00:07:15', '2016-11-26T10:21:32', resolution=10)

phobos = spiops.Target('PHOBOS', time=interval, frame='IAU_PHOBOS')
mex = spiops.Observer('MEX', time=interval, target=phobos)
```

spiops

```
mex.Plot('zaxis_target_angle', notebook=True)
```

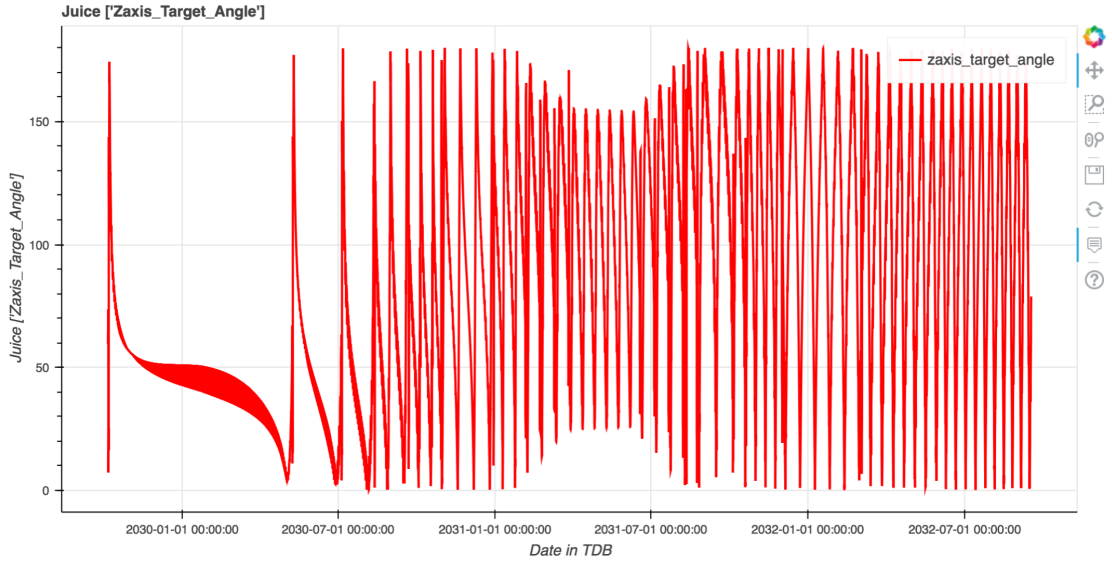
BokehJS 0.12.15 successfully loaded.



spiops

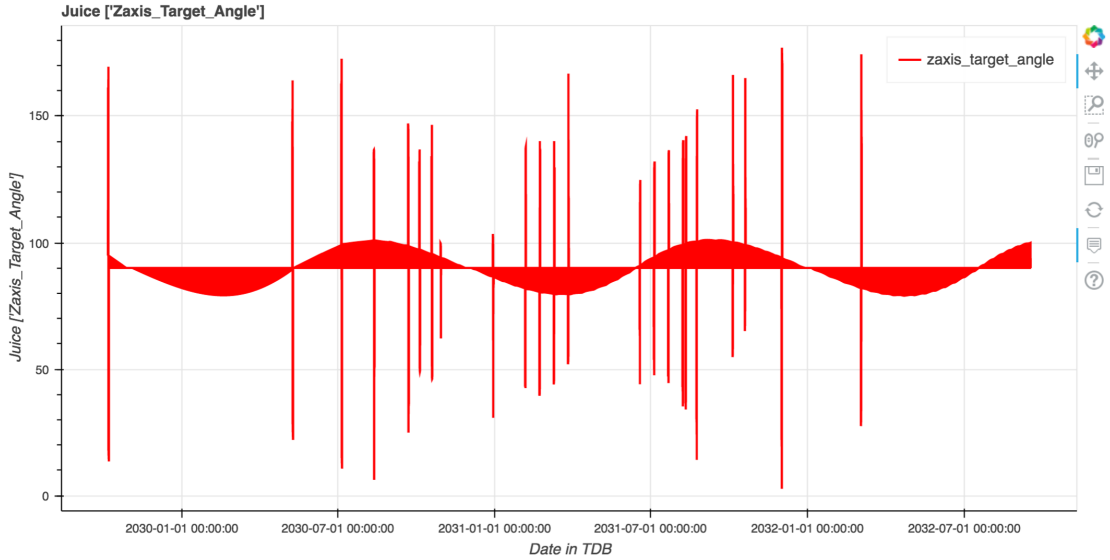
```
interval = spiops.TimeWindow('2029-10-06T00:00:00', '2032-09-16T23:55:00', resolution=60*60)

jupiter = spiops.Target('JUPITER', time=interval, frame='IAU_JUPITER')
juice = spiops.Observer('JUICE', time=interval, target=jupiter)
```

```
juice.Plot('zaxis_target_angle', notebook=True)
```



spiops

```
earth = spiops.Target('EARTH', time=interval, frame='IAU_EARTH')
juice.target = earth
juice.Plot('zaxis_target_angle', notebook=True)
```



spiops

```
import numpy as np

interval = spiops.TimeWindow('2018-06-25T00:00:00', '2018-06-26T00:00:00', resolution=10)

distance = []

timeset = interval.window

for time in timeset:
    state = spiceypy.spkezr('MEX',time,'J2000','NONE','TGO')[0]
    distance.append(np.sqrt(np.power(state[0],2)+np.power(state[1],2)+np.power(state[2],2)))

spiops.plot(timeset, [distance],
           yaxis_name=['Distance [km]'],
           title='MARS-TGO Distance',
           plot_height=500,
           plot_width=900,
           notebook=True)
```

BokehJS 0.12.15 successfully loaded.