Kernel Based Orbit Creation for Uniview

SPICE Kernels are the data files used by NASA to monitor the position of spacecraft, satellites, asteroids, and other objects in our solar system. The name is an acronym, breaking down the data stored in the kernel: Spacecraft ephemeris, Planet ephemeris, Instrument description, C-matrix position information for instruments and Events. Although this can provide a lot of interesting data, in general, for the purposes of Uniview, we might only worry about spacecraft and planet/other body ephemeris to position it in our simulation of the solar system.

Many SPICE kernels can be obtained directly from NASA via the Navigation and Ancillary Information Facility<http://naif.jpl.nasa.gov/naif/index.html>. NAIF has kernels for several active missions and major bodies, and is also the source of the SPICE toolkit, a selection of command line utilities to work with SPICE data, available for windows, mac & linux <http://naif.jpl.nasa.gov/naif/utilities.html>. However, not all of NAIF's SPICE kernels are current, as kernels are restricted under ITAR, the International Traffic on Arms Regulation. Although the kernels themselves are often restricted and need to be cleared before being released, the raw trajectory and ephemeris information, and kernels created by citizens, are not considered such risks, having been stripped of a significant portion of the data. NASA and NAIF provide instructions on kernel creation to everyone with access to the page, although I am summarizing them here.

To obtain data for a spacecraft or body not available on NAIF (or that only has incomplete information available), I will often use JPL HORIZONS. HORIZONS will poll kernels and other solutions, even when they are not available to the public and provide data that can be reconstituted into a new SPICE kernel.

HORIZONS can provide many functions, and getting data to construct a SPICE kernel requires a somewhat different set-up than usual. For this example, I'll be using the Dawn spacecraft (although there is already a module on UCare using a kernel from NAIF). For these purposes, I'll often use the HORIZONS web interface at<http://ssd.jpl.nasa.gov/horizons.cgi> although it is also possible to generate kernel information from the telnet interface. To get started, you'll need to change several settings from their defaults.

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/62974.jpg |
| Click for full-size image |

To get trajectory data, you need to change your Ephemeris Type to VECTORS. By default, it will be Observer Table, and if you were seeking Keplerian Orbit information, it would have been Orbital Elements.

Next, I'll select the TARGET BODY

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/b89e5.jpg |
| Click for full-size image |

Keep in mind, there are a lot of asteroids, and several reused names between spacecraft and asteroids. In addition, sometimes the common name for a mission may not be exactly the name it has on HORIZONS, so it can take a little searching. Sometimes launch vehicle parts will also have a kernel, so make sure you select the spacecraft itself.  
  
Next, set the coordinate origin. This menu can be a little confusing, because the search function is really set up to search locations on the Earth for creating observer tables.

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/33dba.jpg |
| Click for full-size image |

There are instructions below the search function though. In general, I use 500@0 to get the Solar System Barycenter, but you might also use 500@10 (keep in mind, SSB and Sun are NOT the same). The planets advance in their expected order, ie, 500@1 is Mercury Barycenter, 500@2 is Venus Barycenter, etc.  
  
Next, set your timespan. Most kernels are only available for a certain range. Luckily, once you've selected your spacecraft it will usually tell you the range of dates available, so you can just copy them in. If you get an error, try adding or subtracting a few minutes or hours from the values given.

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/284c0.jpg |
| Click for full-size image |

You can also set your step size, which will depend on the precision you need. Keep in mind, the smaller the steps, the larger your final file, and for long missions this won't always be necessary. Use discretion. Horizons will also only generate about 90,000 lines of data, so if you exceed this, you'll need to split up your query into two or more parts (which you can do with the start/stop time).

Finally, you need to have specific table settings.

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/f3d7b.jpg |
| Click for full-size image |

I've had the most success with the settings above: output units should be AU & AU/d (you can specify KM & KM/d, but I've had issues with mkspk (covered later) interpreting KM/d. AU/d seems to provide sufficient precision, though). Quantities must be 2 (state vector {x, y, z, vx, vy, vz}). By default HORIZONS will only give you x, y & z, but you need the vx, vy & vz to get your trajectory! Reference plane should be ecliptic & mean equinox. It is possible to use body equatorial system, but you are unlikely to need this. Reference system should be ICRF/J2000.0, you want Geometric States, and I like CSV format output.

Finally, hit Generate Ephemeris and Horizons will spit out your results. It may take a couple of minutes for a big request. It will usually provide some suggestions if it has an error. Once done you'll see the information about the spacecraft and mission, followed by a great big table of data. Select EVERYTHING on the table between $$SOE...

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/030d6.jpg |
| Click for full-size image |

And $$EOE

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/540bd.jpg |
| Click for full-size image |

And paste your results into a blank text document. I like to use notepad++ (or textrwrangler on a mac), rather than just notepad.

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/d3c36.jpg |
| Click for full-size image |

This will become our csv file which will act as a reference for our body. In case you didn't notice above, the values are as follows for each line: Julian Ephemeris Date, Human-readable calendar date, x-coordinate, y-coordinate, z-coordinate, x-velocity, y-velocity, z-velocity.

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/e0f2a.jpg |
| Click for full-size image |

The next step is one I do personally, but may not be necessary. I just go through and replace " A.D. " (space A.D. space) with " " (blank space). Since MOST data I'm working with is in just the past few and next few decades, and Uniview only really handles those dates anyway, I rarely need anything earlier and want to avoid confusing the utilities.

Then save your file. You want to save it as *something*.csv. Since this is for dawn, I saved it as dawn.csv. I keep SPICE data, especially my working files, all in a folder in the root of my C: called "spice." Just to make my life easy.

Next you need to create a so-called set-up file.

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/44c8c.jpg |
| Click for full-size image |

[ownload file "dawntraj.txt"](http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/attachments/36a05/dawntraj.txt)

It's important to fill out this set-up file right! Don't skip anything important. I want to run through the values:

Input data must be 'STATES' (geometric states), and the INPUT\_DATA\_FILE should be the .csv file you just made.  
There are over a dozen OUTPUT\_SPK\_TYPEs, but the most useful is type 9, LaGrange interpolation with uneven states (this is best because you don't have to think about leap seconds, which HORIZONS automatically used). Type 9 is, in fact, intended for general use.  
OUTPUT\_SPK\_FILE is the name of the file you want to make, here, 'dawn.bsp' note that mkspk will never overwrite an existing file!  
The OBJECT\_ID is a number you should be able to find on HORIZONS--if it didn't show up right next to your target body name, it will usually be in the first few lines generated on the output page, or can sometimes be looked up elsewhere. Spacecraft usually start with a "-" whereas planets, comets and asteroids don't. Dawn is -203. For object name I entered 'DAWN SPACECRAFT' however mkspk knows the right name from -203, and may override this if I entered it wrong. CENTER\_ID and CENTER\_NAME work the same, but for your center.  
My reference frame is 'ECLIPJ2000'--the ECLIP is important, if I just put in 'J2000' it will assume I wanted equatorial coordinates, and tilt my entire orbit by 23.5 degrees (unless I had an equatorial source, like a satellite around Earth might be).  
TIME\_WRAPPER is '# UTC'--numerical UTC time.   
PRODUCER\_ID is you!  
DATA\_ORDER is the order it should read the items in each line, which may be different depending on how you made your kernel file. I have it set to 'skip epoch x y z vx vy vz'--but because the first two items are JED and human-readable date, I also could have done 'epoch skip x y z vx vy vz.' If I had additional data, I might need something like 'epoch skip x y z vx vy vz skip skip skip'.

INPUT\_DATA\_UNITS should be self evident, angles in degrees, distances in AU and velocity in AU/D. It will default to these values if there's an error... in fact, you can see I misspelled 'VELOCITY' in the image above which is probably why I used to have trouble getting KM/D to work...  
LINES\_PER\_RECORD is 1, but again, if you generated it differently, you might have multiple lines.  
LEAPSECONDS\_FILE is a file obtainable from NAIF that contains a list of all leapseconds. The most recent is naif0010.tls. Enter this to generate your data correctly. Finally, POLYNOM\_DEGREE is 9 and SEGMENT\_ID is SPK\_STATES\_08.

Once your trajectory file is built, go ahead and save it. I always save it as just .txt, but I don't really think mkspk cares about the filetype for the set-up file.

For the next step, you need the spice utilities. I recommend at least getting mkspk.exe and spacit.exe. I keep these executables in my C:/spice folder, with all my data. Browse to your directory and run mkspk. On windows, use the command prompt, mac or linux, your terminal.

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/7ef2e.jpg |
| Click for full-size image |

Type in the name of your setup file (keep in mind you'll need to type out the directory if it lives elsewhere... see why I keep everything together?).

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/c9791.jpg |
| Click for full-size image |

mkspk will take a few seconds to a few minutes to work, writing all the data from your .csv file into a binary SPICE kernel (.bsp) file, as specified in your set-up file.

Because bsp files are binary, you can't just open them and read them. This is where you can use spacit to check your file (it can also convert into human-readable SPK transfer files, and back into bsp), or, if you got your file fright from NAIF, to get the necessary information out of the file. Load it and tell it to summarize a binary file.

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/36c55.jpg |
| Click for full-size image |

It will then ask you for the bsp file you want to read, and the leapseconds file to use.

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/9d516.jpg |
| Click for full-size image |

Same as before!

Now it will tell me the details of my file. Longer flights may have multiple steps for their trajectory, so you may get several results printed out with different date ranges.

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/2d707.jpg |
| Click for full-size image |

Once you are happy, I suggest leaving this open, but you can quit spacit. Now copy your .bsp file over to the custom module folder of the trajectory you want to build.

First, let's edit the .mod file.

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/010ee.jpg |
| Click for full-size image |

There's a lot going on here. First of all, Uniview works off of these "coordinate systems" which are like spherical bubbles in space which tell it where objects are and how the camera should behave. My first one here is one I'm calling "c\_dawn"--coordinate system for dawn, which is a child of the SolarSystem. My unit is 1000km (pretty common for SolarSystem stuff), and entrydist is 500... which is 500xmy unit, or 500,000km. Here I specify my position file, orbit file and the position hook, which must be the actual object I create.

I then make a second coordinate system, which I use to SCALE the spacecraft. If I have a model, Uniview will interpret the units of my 3D program as whatever the unit for the lowest coordinate system is. One can use this to very precisely scale objects (although small objects nearby big ones can experience precision errors, depending on how their coordinate systems overlap). In this case, I made it a mere 10 meters. Note that this coordinate system is a child of the first one.

Finally, I make the object itself. In this case, I accidentally set the scale twice, and Uniview will use the lowest one. Here I also specify the model file to use, the target radius (how far away from the craft I will stop when I fly to or jump to it) and the cameraradius (how far away the invisible wall around the object is to prevent me from flying into it) and finally, the guiname. Keep in mind, coordinate systems, objects and guinames should not conflict with existing names! This can cause issues.

Now let's edit the position.conf file.

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/3da16.jpg |
| Click for full-size image |

Note the layout of the file. It's also possible (and sometimes, necessary!) to make a file that contains both a spice AND keplerian orbit, in case the spice kernel expires, or the orbit is very long (such as with the outer dwarf planets--see the dwarf planets module for reference). Note what I specify here: the body, either by body ID number, or I can use the name I see in SPACIT (which may be different from the name listed on HORIZONS or the one entered in my set-up file, in this case, I could say either -203 OR DAWN). Then my observer, which is either 0 or SOLAR SYSTEM BARYCENTER (keep in mind, SUN is different! If I enter the wrong observer, Uniview will draw my orbit centered incorrectly which will throw everything off!). Next is scale, in this case, 0.00001. Honestly, I'm not quite sure why this is the exact right value to use, but it is. This is, when your parent coordinate system is the Solar System, the right value. If your parent coordinate system is instead orbiting a planet (without a larger reference frame), you will instead need to enter .0014959 which will convert between AU and KM. You can empirically check your values by opening Uniview and seeing if you're the right distance. Finally, you can specify the kernel, the file you created (or got straight from NAIF) and even the leapseconds file.

That done, save it and move on to the orbit.conf file.

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/e211f.jpg |
| Click for full-size image |

There are two kinds of orbits: open (like a spacecraft flying out into space) and closed (a spacecraft orbiting another body). SPICE kernels can be associated with either--for example, Spitzer or Kepler can be driven by a spice kernel, but have a closed trajectory around the sun. For closed trajectories, you mostly just need to know the orbital period (which HORIZONS can give you). For open, though, you need to know the date range for which your trajectory is valid--once again, information SPACIT will give you (of course, it's the same as the date range you started with). Sometimes HORIZONS will have a day or two of excess (or lack of data) on either side of the trajectory, so you can also trim it here if you get funky artifacts like the orbit line drawing into the Sun (when SPICE kernels expire, or before they begin, behavior can be unpredictable: sometimes Uniview will draw the object at its starting position, or final position, or just dump it inside the sun, or at the solar system barycenter).

That done, add your module to Uniview and check it out!

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/27f93.jpg |
| Click for full-size image |

You can check the date range and make sure your spacecraft leaves Earth and follows its orbit as intended. Here we can see Dawn flying away from Earth, following along with Vesta, then leaving for Ceres. Note, I had to use a kernel driven Vesta and Ceres too, because Keplerian orbits aren't perfect, especially the further out you are from the date of elements. With the built-in keplerian Ceres, Dawn arrived a couple hours too early for their encounter (the complete module, with a better date range than these examples, is on UCare).

|  |
| --- |
| http://aveewiki.calacademy.org/groups/planetariumdocuments/wiki/76834/images/__thumbs__/e680e.jpg |
| Click for full-size image |

Now you can enjoy high-precision flightpaths and kernels for Uniview!