A MATLAB Implementation of the JPL Ephemeris

This document describes two MATLAB functions and companion scripts that demonstrate how to read JPL binary ephemeris files and calculate the position and velocity vectors of a planet, the sun or the Moon. The first function, `jplephem.m`, was ported to MATLAB using the Fortran source code subroutine provided by JPL. This function reads and evaluates binary ephemeris files created using the JPL-provided ASCII data files. These ASCII data files and Fortran source code can be found at ftp://ssd.jpl.nasa.gov/pub/eph/planets/ascii/. A CD ROM containing these ASCII data files and the companion Fortran source code is also available from Willmann-Bell, www.willbell.com.

The second function, `jpleph_mice.m`, uses routines from the MICE software suite to read and evaluate binary ephemeris files available at naif.jpl.nasa.gov/naif/toolkit_MATLAB.html. MICE is a MATLAB implementation of the SPICE library created by JPL.

Each ephemeris function requires initialization the first time it is called. The following three statements in the main MATLAB script will perform the proper initialization for `jplephem`:

```
iephem = 1;

ephname = 'de405.bin';

km = 1;
```

The initialization for the `jpleph_mice` function is as follows:

```
iephem = 1;

ephname = 'de405.bsp';

km = 1;
```

The second item is the name of the binary ephemeris file to use for all calculations. The third item determines the units of the output. If `km = 1`, the output will be in the units of kilometers and kilometers per second. If `km = 0`, the output will be in the units of AUs and AUs per day. These three items should be placed in a `global` statement at the beginning of the main script which calls either of these functions.

Note that the value of the Astronomical Unit, in kilometers, used in a particular JPL ephemeris is available as the constant `au` which is placed in global by the `jplephem` function. However, when using the `jpleph_mice` function, the user must define this numerical value in the main script and also make it available in a global statement. The actual value used in a particular JPL ephemeris can be found in the header file posted on the JPL website. For example, the following are the first three lines in the `header.421` data file. The value of Astronomical Unit used in this ephemeris is the first number in row three of this data file.

```
0.42100000000000000D+03  0.42100000000000000D+03  0.00000000000000000D+00
0.12008021181181170000D+17  0.00000000000000000D+00  0.29979245800000000D+06
0.14959787069962620D+09  0.81300569069915300D+02  0.49125495718679400D-10
```

`demo_jpl` is a simple MATLAB script that demonstrates how to interact with the `jplephem` function. Binary ephemeris files for Windows compatible computers can be downloaded from the Celestial and Orbital Mechanics web site located at www.cdeagle.com.

The coordinates calculated by this function are with respect to the International Celestial Reference System (ICRS) which is described in "The International Celestial Reference Frame as Realized by Very Long Baseline Interferometry", C. Ma, E. Arias, T. Eubanks, A. Fey, A. Gontier, C. Jacobs, O. Sovers, B. Archinal and P. Charlot, *The Astronomical Journal*, 116:516-546, 1998, July.

The following is the syntax for the `jplephem` MATLAB function:

```
function rrd = jplephem (et, ntarg, ncent)

% reads the jpl planetary ephemeris and gives
% the position and velocity of the point 'ntarg'
% with respect to point 'ncent'

% input

%   et    = julian ephemeris date at which interpolation is wanted

%   ntarg = integer number of 'target' point

%   ncent = integer number of center point

%   the numbering convention for 'ntarg' and 'ncent' is:

%         1 = mercury           8 = neptune
%         2 = venus             9 = pluto
%         3 = earth            10 = moon
%         4 = mars             11 = sun
%         5 = jupiter          12 = solar-system barycenter
%         6 = saturn           13 = earth-moon barycenter
%         7 = uranus           14 = nutations (longitude and obliq)
%                              15 = librations, if on ephemeris file

%        if nutations are wanted, set ntarg = 14.
%        for librations, set ntarg = 15. set ncent = 0.

% output

%   rrd = output 6-word array containing position and velocity
%         of point 'ntarg' relative to 'ncent'. the units are au and
%         au/day. for librations the units are radians and radians
%         per day. in the case of nutations the first four words of
%         rrd will be set to nutations and rates, having units of
%         radians and radians/day.

%         the option is available to have the units in km and km/sec.
%         for this, set km = 1 via global in the calling program.
```

The following is a typical user interaction with this script using the DE421 ephemeris file.

```
demo_jpl - demonstrates how to use the jplephem.m function

please input a UTC calendar date

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 10,21,1998

    target body menu

  <1> Mercury
  <2> Venus
  <3> Earth
  <4> Mars
  <5> Jupiter
  <6> Saturn
  <7> Uranus
  <8> Neptune
  <9> Pluto
  <10> Moon
  <11> Sun

please select the target body
? 10

    central body menu

  <1> Mercury
  <2> Venus
  <3> Earth
  <4> Mars
  <5> Jupiter
  <6> Saturn
  <7> Uranus
  <8> Neptune
  <9> Pluto
  <10> Moon
  <11> Sun
  <12> solar-system barycenter
  <13> Earth-Moon barycenter

please select the central body
? 3
```

The following is the MATLAB script output for this example.

```
program demo_jpl


ephemeris file     de421.bin

target body        'Moon'

central body       'Earth'


UTC calendar date       21-Oct-1998
```

```
UTC Julian date          2451107.50000000

TDB Julian date          2451107.50073128


state vector

        rx (km)                 ry (km)                 rz (km)                rmag (km)
 -3.37366639992093e+005  -2.19024588322606e+005  -5.98176034774959e+004  +4.06652389339142e+005

        vx (kps)                vy (kps)                vz (kps)               vmag (kps)
 +5.39083723067669e-001  -7.55355631361622e-001  -2.86179850104219e-001  +9.71119095122677e-001
```

Note that UTC is the Coordinates Universal Time and TDB is the Barycentric Dynamical Time.

The following is the syntax for the `jpleph_mice` MATLAB function:

```
function rrd = jpleph_mice (et, ntarg, ncent)

% reads the jpl planetary ephemeris and gives the position and velocity
% of the point 'ntarg' with respect to point 'ncent' using MICE routines

% input

%   et    = TDB julian date at which interpolation is wanted

%   ntarg = integer number of 'target' point

%   ncent = integer number of center point

%   the numbering convention for 'ntarg' and 'ncent' is:

%         1 = mercury           8 = neptune
%         2 = venus             9 = pluto
%         3 = earth            10 = moon
%         4 = mars             11 = sun
%         5 = jupiter
%         6 = saturn
%         7 = uranus

% output

%   rrd = output 6-word array containing position and velocity
%         of point 'ntarg' relative to 'ncent'. the units are
%         determined by the value of km passed via global.

% global

%   iephem  = initialization flag (1 = initialize)
%   ephname = name of ephemeris binary data file (de421.bsp, etc.)
%   km      = state vector units flag (1 = km & km/sec, 0 = au & au/day)
%   au      = numerical value of astronomical unit (kilometers)
```

`demo_jpl_mice` is a simple MATLAB script that demonstrates how to interact with the `jpleph_mice` function. Binary ephemeris files for can be downloaded from the JPL NAIF website located at naif.jpl.nasa.gov/naif/toolkit_MATLAB.html.

The following is a typical user interaction with this script using the DE421 ephemeris file.

```
demo_jpl_mice - demonstrates how to use the jpleph_mice.m function

please input a UTC calendar date

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 10,21,1998

    target body menu

  <1> Mercury
  <2> Venus
  <3> Earth
  <4> Mars
  <5> Jupiter
  <6> Saturn
  <7> Uranus
  <8> Neptune
  <9> Pluto
  <10> Moon
  <11> Sun

please select the target body
? 10

    central body menu

  <1> Mercury
  <2> Venus
  <3> Earth
  <4> Mars
  <5> Jupiter
  <6> Saturn
  <7> Uranus
  <8> Neptune
  <9> Pluto
  <10> Moon
  <11> Sun

please select the central body
? 3
```

The following is the MATLAB script output for this example.

```
program demo_jpl_mice


ephemeris file     de421.bsp

target body        'Moon'

central body       'Earth'


UTC calendar date       21-Oct-1998

UTC Julian date          2451107.50000000
```

```
TDB Julian date          2451107.50073128


state vector

        rx (km)                 ry (km)                 rz (km)                rmag (km)
 -3.37366639992093e+005  -2.19024588322606e+005  -5.98176034774959e+004  +4.06652389339142e+005

        vx (kps)                vy (kps)                vz (kps)               vmag (kps)
 +5.39083723067670e-001  -7.55355631361621e-001  -2.86179850104219e-001  +9.71119095122677e-001
```

It is good programming practice to close the binary ephemeris file at the end of the main script with the following statement:

```
% unload ephemeris

cspice_unload('de421.bsp');
```

Please note the following extracted from JPL IOM 343R-08-003, dated 31-March 2008.

In a resolution adopted by the International Astronomical Union in 2006 (GA26.3), the time scale TDB (Temps Dynamique Barycentrique, Barycentric Dynamical Time) was defined to be consistent with the JPL ephemeris time.

The axes of the ephemeris are oriented with respect to the International Celestial Reference Frame (ICRF).

Furthermore, the relationship between the ICRF and EME 2000 coordinate frames is described in the `moon_080317.tf` text file which is located on the JPL ftp server at

ftp://naif.jpl.nasa.gov/pub/naif/generic_kernels/fk/satellites/,

The IERS Celestial Reference Frame (ICRF) is offset from the J2000 reference frame (equivalent to EME 2000) by a small rotation; the J2000 pole offset magnitude is about 18 milliarcseconds (mas) and the equinox offset magnitude is approximately 78 milliarcseconds.

The following describes a MATLAB function that can be used to compute the J2000-to-ICRS transformation matrix.

j2000_icrs.m – J2000-to-ICRS transformation matrix

This MATLAB function returns the transformation matrix from the mean dynamical equator and equinox at J2000 to the International Celestial Reference System (ICRS). There are two options for this matrix based on different data types via the `itype` argument.

The following is the syntax for this MATLAB function:

```
function tmatrix = j2000_icrs(itype)

% transformation matrix from the mean dynamical
% equator and equinox at j2000 to the
% International Celestial Reference System (ICRS)

% input
```

```
%  itype = type of transformation
%      1 = LLR + VLBI
%      2 = Chapront et al. LLR

% output

%  tmatrix = transformation matrix

% reference: Rotation Matrix from the Mean Dynamical
% Equator and Equinox at J2000.0 to the ICRS, Astronomy
% and Astrophysics, October 1, 2003.
```

To transform coordinates from the ICRS system to the J2000 system, the programmer should use the transpose of the matrix returned by this function.

Leap seconds calculation

The difference between International Atomic Time (TAI) and Universal Coordinated Time (UTC) is the number of current leap seconds. International Atomic Time (TAI**,** Temps Atomique International) is a physical time scale with the unit of the SI (System International) second and derived from a statistical timescale based on a large number of atomic clocks. Coordinated Universal Time (UTC) is the time scale available from broadcast time signals. It is a compromise between the highly stable atomic time and the irregular earth rotation. UTC is the international basis of civil and scientific time.

The calculation of leap seconds in these MATLAB scripts is performed by a function that reads a simple ASCII data file and evaluates the current value of leap seconds. The leap second function must be initialized by including the following statements in the main script.

```
% read leap seconds data file

readleap;
```

The `readleap` MATLAB function reads the contents of the following simple comma-separated-variable (csv) two column data file. The name of this file is `tai-utc.dat`.

```
 2441317.5,  10.0
 2441499.5,  11.0
 2441683.5,  12.0
 2442048.5,  13.0
 2442413.5,  14.0
 2442778.5,  15.0
 2443144.5,  16.0
 2443509.5,  17.0
 2443874.5,  18.0
 2444239.5,  19.0
 2444786.5,  20.0
 2445151.5,  21.0
 2445516.5,  22.0
 2446247.5,  23.0
 2447161.5,  24.0
 2447892.5,  25.0
 2448257.5,  26.0
 2448804.5,  27.0
```

```
 2449169.5,  28.0
 2449534.5,  29.0
 2450083.5,  30.0
 2450630.5,  31.0
 2451179.5,  32.0
 2453736.5,  33.0
 2454832.5,  34.0
```

The first column of this data file is the Julian date, on the UTC time scale, at which the leap second became valid. The second column is the leap second value, in seconds.

Note that this data is passed between the leap second MATLAB functions by way of a global statement.

```
global jdateleap leapsec
```

The MATLAB function that actually reads and evaluates the current value of leap seconds has the following syntax and single argument.

```
function leapsecond = findleap(jdate)

% find number of leap seconds for utc julian date

% input

%  jdate = utc julian date

% input via global

%  jdateleap = array of utc julian dates
%  leapsec   = array of leap seconds

% output

%  leapsecond = number of leap seconds
```

The leap seconds data file should be updated whenever the International Earth Rotation and Reference Systems Service (IERS) announces a new leap second.